NASA
Technical
Paper
3372

August 1993

Neural-Network-Directed Alignment of Optical Systems Using the Laser-Beam Spatial Filter as an Example

Arthur J. Decker,
Michael J. Krasowski,
and Kenneth E. Weiland

NASA

NASA
Technical
Paper
3372

1993

Neural-Network-Directed Alignment of Optical Systems Using the Laser-Beam Spatial Filter as an Example

Arthur J. Decker,
Michael J. Krasowski,
and Kenneth E. Weiland
Lewis Research Center
Cleveland, Ohio

NASA
National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Program

Page intentionally left blank

Contents

Summary

This report describes an effort at NASA Lewis Research Center to use artificial neural networks to automate the alignment and control of optical measurement systems. Specifically, it addresses the use of commercially available neural network software and hardware to direct alignments of the common laser-beam-smoothing spatial filter. The report presents a general approach for designing alignment records and combining these into training sets to teach optical alignment functions to neural networks and discusses the use of these training sets to train several types of neural networks. Neural network configurations used include the adaptive resonance network, the back-propagation-trained network, and the counter-propagation network. This work shows that neural networks can be used to produce robust sequencers. These sequencers can learn by example to execute the step-by-step procedures of optical alignment and also can learn adaptively to correct for environmentally induced misalignment. The long-range objective is to use neural networks to automate the alignment and operation of optical measurement systems in remote, harsh, or dangerous aerospace environments. This work also shows that when neural networks are trained by a human operator, training sets should be recorded, training should be executed, and testing should be done in a manner that does not depend on intellectual judgments of the human operator.

Introduction

This report describes an effort at NASA Lewis Research Center to use neural networks to automate the alignment and control of optical measurement systems. This project, which was supported by the Earth-to-Orbit Propulsion Instrumentation Working Group of NASA, was begun in 1989 because of a need to make optical measurements near an operating test bed of the Space Shuttle Main Engine where the environment is intolerable for humans.

Aerospace measurement environments can be characterized in terms of two challenges. The first challenge is the optical access of areas of interest in the experiment, rig, or facility. The second challenge, the one that we address in this report, is that hands-on alignment, adjustment, and control is often difficult or impossible. Although hands-on adjustment of these systems is frequently necessary during a test, human safety considerations often prevent access during testing. This may mean, for example, that a test must be shut down for a period in order to do the required adjustments. Obviously, it would be less costly and far more efficient to automate these adjustments.

For a typical alignment the usual procedure is for a human operator to control the illumination of an extended region visually by using the beam pattern for alignment clues. This procedure is not necessarily trivial. For example, four mirror mounts, each with 3 rotational degrees of freedom, have 12! or 479 001 600 possible orderings of an alignment sequence involving all 12 degrees of freedom. A human operator, of course, imposes many constraints to restrict the number of possible moves. A particular mount can be aligned to center the beam on another mount and then locked. Several moves can be made sequentially to move the hot spot of a beam in a horizontal direction only. Nevertheless, even a simple optical measurement system may require frequent random adjustments of 3 to 6 degrees of freedom. A recent test of the simplest off-axis reference-beam holography setup, with components already laid out, required between 50 and 100 translational and rotational motions to bring the setup into alignment.

Solving the problem of automated alignment then requires automating the learned human skill of pattern-directed operation of a complex system of controls. An approach to this problem has recently become available. This approach involves a method of parallel processing referred to as an artificial neural network (ref. 1). As will be discussed later, an artificial neural network can learn to map a general set of input patterns into an appropriate set of patterns of output control actions.

The advantage of using a neural network is that, like a human operator, it can learn an alignment procedure, a control law, or any other mapping by example. It is not necessary to discover a mathematical representation of the mapping by human analytic processes. The human operator need only know by experience a representative set of input patterns and output control information. Such a set of patterns and information is called a training set. However, the procedure is not quite that straightforward: the person training the neural network must know or discover the composition of the input pattern. Part of this work consisted of discovering an optical alignment paradigm (ref. 2).

Application of neural networks to the optical alignment problem consisted of several tasks:

(1) Study of the theory of neural networks and their application to optical pattern recognition (refs. 3 and 4)
(2) Acquisition of neural network development systems (refs. 5 to 7)
(3) Selection of a benchmark component for automated alignment—the spatial filter commonly used for laser beam smoothing and signal isolation (ref. 8)
(4) Generation of human-directed alignment records to use for training neural networks or several systems of neural networks
(5) Testing the systems of neural networks—comparing neural-network-directed alignments of the spatial filter with human-directed alignments

The spatial filter was chosen as a benchmark component because it is a simple component whose alignment is pattern controlled. Its theory, models, and pattern visualization were important for this effort, but the neural network procedures and alignment paradigm that resulted are considered to be quite generally applicable.

The types of neural networks used included the back-propagation-trained network (BPN) (ref. 9), the counter-propagation network (CPN) (ref. 10), the Euclidean Preclassifier (ref. 11), and the Adaptive Resonance Technique 2 (ART2) (ref. 12).

This report discusses these networks in the order mentioned. The theory of the laser-beam-smoothing spatial filter is given in appendixes A and B, and a list of symbols is given in appendix C.

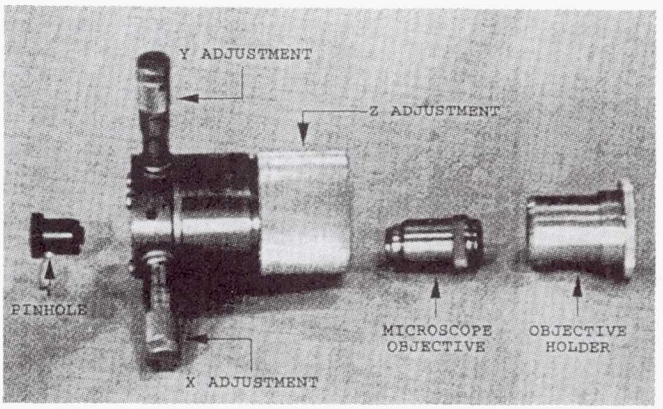


Figure 1.—Disassembled spatial filter assembly.

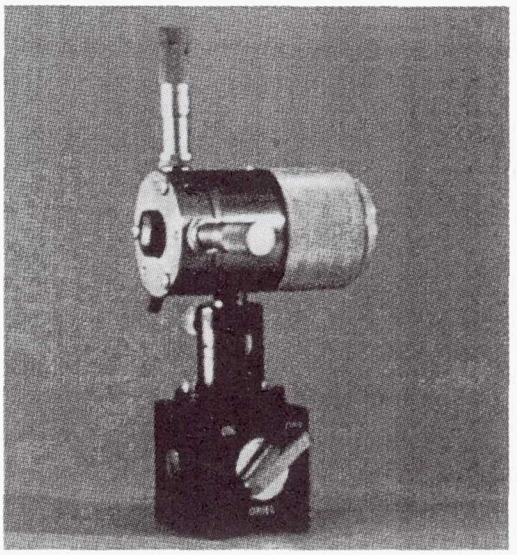


Figure 2.—Spatial filter assembly.

Automation of Optical System Alignment

Description of Alignment Process

The spatial filter (fig. 1) and its alignment are understood, in principle, from the science of physical optics. Figure 1 is a photograph of a disassembled spatial filter of the type used to clean up laser beams in the laboratory. Figure 2 shows the assembled spatial filter, which consists of a microscope objective (typically 20×) that focuses a laser beam onto a pinhole (typically 10 μm in diameter). Scattered light (from dust particles on the lenses, for example) does not generally pass through the pinhole. An aligned spatial filter thereby filters the scattered light from the laser beam.

A person learns how to align these spatial filters with some practice. There are two alignment procedures required: (1) centering the laser beam on and making it coaxial with the microscope objective and (2) aligning the pinhole with the microscope objective. The first procedure, which is done with the pinhole removed, can be complex; it may require the adjustment of several mirrors, prisms, or beam elevators. Only the second procedure is considered for automation in this report.

For the second procedure, the pinhole is inserted, and the laser beam is centered and focused onto it. The spatial filter assembly (figs. 1 and 2) supports the pinhole in an X–Y translation stage with micrometer adjustments. The microscope objective is in a separate Z-axis stage for focusing. This Z-axis stage is the large, knurled cylinder shown in figures 1 and 2. These three controls allow the focal spot (beam waist for gaussian laser beams) to be centered on the pinhole. If the first alignment procedure has been done correctly, the X, Y, and Z controls can independently adjust the X, Y, and Z coordinates of the focal spot relative to the center of the pinhole. On rare occasions, the total Z-motion might be as large as 1000 μm, and the X- or Y-motion as large as 300 μm for the

spatial filter containing a 20× microscope objective and a 10-μm pinhole. Final adjustments, however, might be as small as 1 μm.

The human approach to executing this second procedure varies quite a bit. Appendix A contains the theoretical interpretation of the beam pattern observed. For the following description, which we believe is typical, albeit inexact, figure 3 shows the appearance of the laser beam at the various stages of alignment.

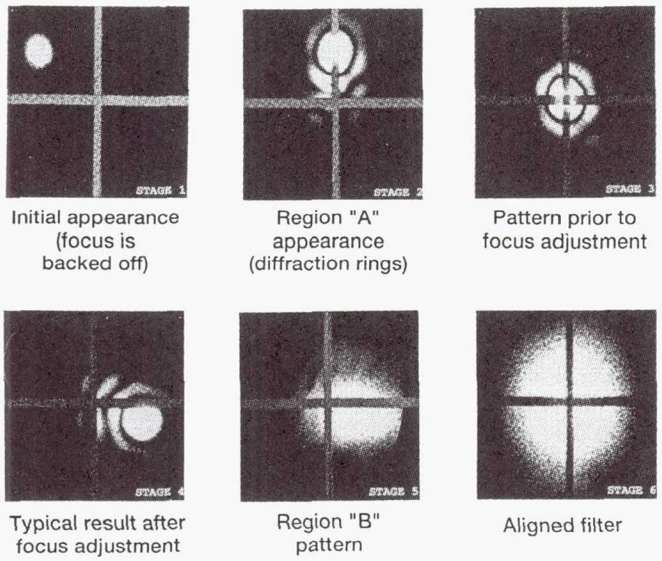


Figure 3.—Appearance of filtered laser beam at selected stages of alignment process for spatial filter.

First, the room lights are turned off, and the focus is backed off toward the laser. This process allows the laser beam to fill the pinhole. Then the output of the filter is projected onto a white card for viewing. Internal reflections can occur, and faint multiple beams might be observed. The main beam will have a bright spot and a pattern of concentric rings, but initially this pattern may be too faint to be seen. In fact, internal reflections may produce beams that are brighter than the main beam. An experienced person can still tell the difference between the main beam and the internal reflections. The region A model of appendix A is an attempt to describe the pattern of this main beam (region A and region B are used to designate regions of alignment space). Typically, the human operator begins centering the main beam by moving the X or Y control the same direction that the beam is to move. Usually, the largest X or Y error is corrected first. The beam usually brightens considerably during these corrections, and the ring pattern becomes visible. Internal reflections become negligible.

Once the beam is centered, the focus is then moved toward the viewing card. The beam's reflection from the card normally will move away from the center during this process, unless the pinhole is exactly centered on the beam. In addition, the beam normally will become fainter. The focus or Z-axis control is operated until the beam is near the edge of the field of view or until the beam starts becoming too faint.

The centering procedure is then repeated with the X and Y controls followed by the focusing procedure with the Z-control. At some point, the appearance of the beam will change significantly. The pronounced ring pattern may disappear, the beam may become asymmetrical, the central spot may appear to be larger and softer, or the beam may brighten substantially. The region B model of appendix A represents this behavior. Essentially, in this region the beam is too small to fill the pinhole uniformly. Although the spatial filter is almost aligned, the beam patterns vary much more for a given control change than they do in region A. In region B, beams that are apparently off center or asymmetrical require slight adjustments in X or Y. Beams that are essentially circular require slight adjustments of Z. An experienced person learns to recognize the action required by a particular beam pattern. At some point, the beam will appear the same as an unfiltered gaussian laser beam. The filter is then aligned, and the human operator stops making adjustments.

The task of automating the alignment of an optical system can be approached in various ways. In each, a given input (expressed as a vector including the present state of the system, past alignment actions, and estimates of future output) produces the output (expressed as a vector including control settings) that is used to direct the next action by the effector, which can be human or electromechanical. The object is to always move closer to the aligned state. The process of examining the current state of alignment and performing an appropriate alignment step is repeated until the alignment is complete.

One way to automate the alignment process would be to model the process theoretically. Such a model might be used directly in a control system as a transfer function. Another approach would be to discover a linear mapping between the whole range of possible input vectors and corresponding output vectors. A third method would be to store the pairwise input-output data in a lookup table. The controller would simply access the desired output control from the table. Severe difficulties exist with all of these methods. The next sections briefly detail the methods and describe the difficulties.

Theories and Models of Spatial Filter Alignment

We have a theory and models of the spatial filter (appendix A) and a model for visualizing the alignment process (appendix B). What prevents us from developing alignment control systems from these theories and models alone? There are many reasons, but the key words are "credibility" and "practical difficulty." Some of the reasons follow:

(1) The models are based on a diffraction integral theory that ignores coupling between components of the electromagnetic field.

(2) The diffraction integrals themselves are simplified to a paraxial or small-angle approximation.

(3) The spatial filter model is assumed to have thin lenses, zero-thickness apertures, and loss-free, reflection free, internal surfaces.

(4) The theory (eqs. (A1) to (A11)) is ill-posed. The beam pattern is computed from the control information (misalignment coordinates), whereas the objective is to obtain control information from the beam pattern.

(5) The models are approximations of the theory itself. Two distinct alignment regions are proposed, and the nature of the transition between the two is ignored.

(6) The numerical calculations from the models are complicated and require significant computer time.

(7) The application of the theory assumes linear detectors (the beam patterns may differ, although the detector readouts are the same); hysteresis, backlash, and other nonlinear mechanical errors are not considered.

This discussion applies to the spatial filter, but it is characteristic of all attempts to use physical theories and models to understand complex systems. Ill-posedness, numerical complexity, nonlinearities, simplifications, and an incomplete understanding of the human-machine interface greatly devalue this exercise. Nevertheless, some physical understanding of the complex system to be controlled is essential to train the neural networks discussed in the following sections. As an example, we show in appendix B that, theoretically, the laser beam and spatial filter can be made dimensionless. This discovery suggests that the training sets for the neural networks can be supplied in dimensionless form, thereby making the neural networks trained by them applicable to more general laser-beam, spatial-filter combinations.

Linear Mappings and Table Lookup

The process of automating the alignment of a spatial filter can be considered as executing a sequence of nonlinear mappings. Linear mappings and transformations (ref. 13), by contrast, have numerical evaluations represented by the matrix equation

$$\boldsymbol{b} = \boldsymbol{W}\boldsymbol{a} \tag{1}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are column vectors with numbers of components m and n, respectively, and $\boldsymbol{W}$ is an n by m matrix. The use of equation (1) in science or engineering implies a linear system or a system that can be linearized.

One possible solution to a mapping problem is to use a table lookup procedure to compare an input pattern with a representative collection of input patterns and to interpolate in a table of input-output pairs to determine the output control information. The major drawback with table lookup is that entries in the table are considered to be independent. The training and memory requirements can be quite large. As an example, consider one of the laser-beam, spatial-filter combinations used for the work reported herein.

This combination consisted of a 30-mW helium-neon laser and a spatial filter with a 20× microscope objective and a 10-μm pinhole. As stated before, the total ranges of the X, Y, and Z mechanical motions were contained in a volume approximately 300 μm by 300 μm by 1000 μm. The mechanical resolution in X, Y, or Z was about 1 μm, so there were about 90×10^6 distinct positions. Each position had a beam pattern. Labeling the beam patterns based on the 1024-pixel, 8-bit characterizations of the beam patterns (the resolution used in appendix B for visualization) would require 92.16 GB of address space. Storing, for example, a 2-bit representation of the control to be selected for operation (X, Y, or Z) would then require an additional 23.04 GB of storage space. For adequate table lookup results, the whole range of input space would have to be covered with the stored alignment records.

The obvious objections to this method are that no human operator would ever collect 90×10^6 training examples for any system and that memory requirements are unacceptably large.

The table lookup method can be modified to use less data. The modifications, which are described in terms of a lookup table space, wherein input vectors are called points, follow:

(1) Eliminate points that have zero probability.

(2) Divide the space into volume elements that are sized to be visited with equal probability.

(3) Determine an exemplar for each volume element.

(4) Develop a lookup algorithm that associates a point with its exemplar.

(5) Read out the output pattern associated with the exemplar.

(6) Interpolate between outputs, particularly if a point has a nonzero probability of being associated with more than one exemplar.

The objection to using this procedure is the same as for using theories and models: detailed, specialized knowledge is required.

Artificial Neural Networks

Introduction to neural networks. — Neural networks are a new approach to a very old problem: extracting and implementing the mapping or transformation of a set of input vectors into a set of output vectors. That operation is expressed in functional form as

$$\boldsymbol{b} = f(\boldsymbol{a}) \tag{2}$$

The components of an input vector $\boldsymbol{a}$ might consist of sensor values and results of operations on sensor values from times in the past, components of the output vector $\boldsymbol{b}$ or results of operations on components of the output vector $\boldsymbol{b}$ from times

in the past, and model-derived estimates of future components of $\boldsymbol{b}$. The components of an output vector $\boldsymbol{b}$ might consist of control settings and sensor values. The function f represents a general combination of processes. In a sense, discovering or implementing f is a primary task of both instrumentation and controls personnel. However, measurement errors and noise change the task somewhat: the objective is to discover or to implement an f that minimizes the mean-square-error E between the mapping or transformation and the set of target output vectors. In general, we want to minimize the expression

$$E = \langle \left| \boldsymbol{b} - f(\boldsymbol{a}) \right|^2 \rangle \qquad (3)$$

where < > represents an expectation value.

The discipline of artificial neural networks does not yet have standard terminology (ref. 14). However, neural networks can be viewed as special cases of networks of independent, parallel operating, interconnected processors. The term "parallel distributed processing," which is the title of a fundamental reference (ref. 3) also describes the discipline. This report adopts the operational definition that is shown in figure 4.

Artificial neural networks are biologically inspired (refs. 15 and 16). The terms "synapse," "axon," and "neuron" are used occasionally as in figures 4 and 5. A neuron, or node, has weighted inputs or synapses. In software simulations these weights, by convention, are assumed to belong to the node, and they are part of its local memory. A node sums its weighted inputs. In addition, during training the node can be allowed to add a bias term to that sum. A major change from the linear transformation is that the output or axon value is a nonlinear function of the weighted sum. The important property of the nonlinear function (sometimes called an activation function) is that the network generates internal degrees of freedom. The nonlinear function used in the work

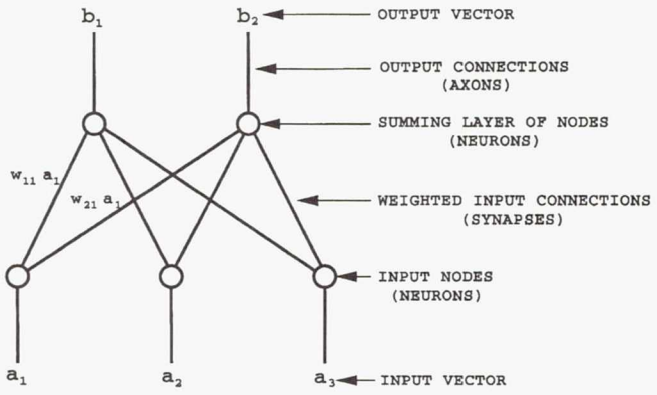


Figure 4.—Linear transformation described by neural network terminology.

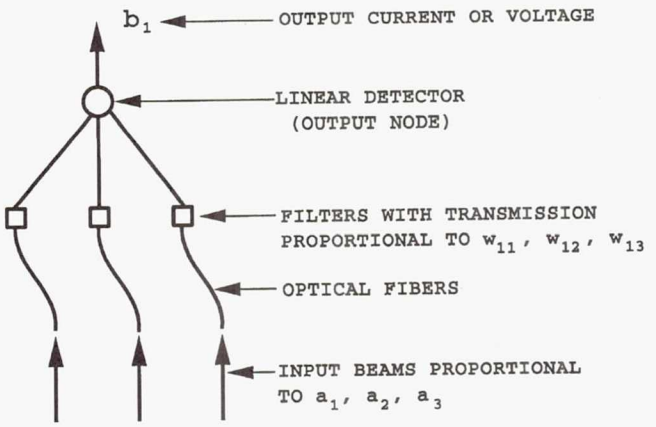


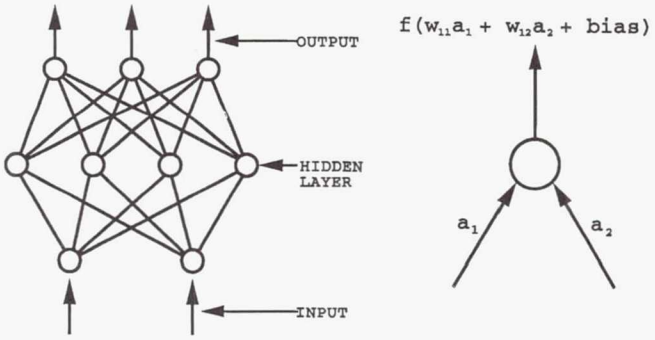
Figure 5.—Concept of optical realization of figure 4. One output node shown.

reported herein is deterministic, or reversible. The output (axon value) usually is transmitted to other nodes, but it also can be fed back as a weighted input to its own node. It can have external inputs for resetting it or for applying external data. Nodes can fire or put out new information synchronously or asynchronously. Interconnections can be fairly arbitrary.

The internal degree-of-freedom generating ability is responsible for the very complex behavior of most nonlinear systems. A nonlinear system with a few input variables acts like a linear system with a much larger number of variables. The nonlinear mechanics or fluid dynamics of these systems is complex because of this phenomenon (ref. 17). Feed-forward networks use this phenomenon to perform the general mappings given by equations (2) and (3). The summations and nonlinear activations constitute an engine, and the weights constitute the knowledge, control information, and program for the engine. Calculating the weights efficiently, accurately, and stably is a primary topic of research and development. This report is concerned with the use of commercially available neural network systems to learn and direct optical alignment. The algorithm used to learn the training set is a weight-calculation algorithm.

The neural-network generalization of the linear mapper discussed previously is called a feed-forward network. Such a network (fig. 6) is arranged in layers of nodes, where a layer receives inputs only from the previous layer and transmits outputs only to the next layer. In addition, unlike the linear mapper, the feed-forward network uses multiple layers. There are a number of modifications of this architecture in use, but the architecture just described allows us to explain the general value of neural networks.

To cause a neural network to learn, the training program presents input and output vectors to the network one pair at a time. For each iteration, the network is allowed to adjust the connection weights using its particular weight-calculation algorithm. The abstract objective is to determine weights $\boldsymbol{W}$



(a) Network.

(b) Sample neuron in hidden layer.

Figure 6.—Feed-forward artificial neural network.

that no longer require adjustment as the entire set of input/output pairs is applied.

Nonlinear differential equations or nonlinear difference equations (for the discrete time case) can be used to describe neural networks as functions of iteration number. Equations (2) and (3) for stable mappings (supposing such mappings exist) should be changed to the equations

$$\boldsymbol{b} = \boldsymbol{g}(\boldsymbol{a}, \boldsymbol{W}) \tag{4}$$

and

$$E = \left\langle \left| \boldsymbol{b} - \boldsymbol{g}(\boldsymbol{a}, \boldsymbol{W}) \right|^2 \right\rangle \tag{5}$$

where $\boldsymbol{W}$ is an array of weights whose values are to be determined by training. If the weights are varied or stepped, a differential equation or difference equation replaces equation (4). The differential equation is given by

$$\frac{d\boldsymbol{b}}{dt} = \boldsymbol{g}\left(\boldsymbol{a}, \frac{d\boldsymbol{a}}{dt}, \boldsymbol{W}, \frac{d\boldsymbol{W}}{dt}, \boldsymbol{b} \right) \tag{6}$$

where both $\boldsymbol{a}$ and $\boldsymbol{W}$ are allowed to vary with the iteration number. When $\boldsymbol{b}$ is fed back as an input to the neural network, it becomes an argument of the nonlinear function $\boldsymbol{g}$.

The problem stated here is an inverse problem; it is generally ill-posed even for linear systems such as those encountered in computed tomography (ref. 18). Fortunately, the accuracy or uniqueness of the weights is not important for the alignment problem as long as they generate correct output vectors.

There are two approaches to developing neural-network architectures and weight-calculation algorithms. The first and oldest approach is phenomenological: take clues from biology and cognitive science (the psychology of learning). This approach is very much aligned with artificial intelligence. An example is the Adaptive Resonance Technique 2 (ART2) that we used in our alignment studies and which we discuss later. References 3 and 15 discuss the psychological and biological viewpoints thoroughly.

The second approach is to apply techniques from the dynamics of nonlinear systems (ref. 17). The most common approach defines an energy function (sometimes referred to as the "cost function" or "Lyapunov function") in terms of the weights. The objective is to design or discover an artificial neural network and energy function where the energy function has minima. Modified steepest-descent techniques are used to determine the weights that correspond to the energy minima. Feed-forward networks are trained by minimizing an energy defined as the mean square error between the generated outputs and the training outputs as in the expression

$$E = \frac{1}{2} \sum \frac{\left(\boldsymbol{b}(\boldsymbol{W}) - \boldsymbol{b}_t \right)^2}{N} \tag{7}$$

where the sum is over the N samples in the training set. The vector $\boldsymbol{b}_t$ is a training vector, and the vector $\boldsymbol{b}$ is the actual output vector resulting from weights $\boldsymbol{W}$ and a training input $\boldsymbol{a}_t$. Topologically, E is imagined to be a surface in a space whose dimension equals the number of weights. The surface has dimples or valleys that represent the minima.

There are two possible complications. The first is that E can have multiple minima, so a solution for $\boldsymbol{W}$ may not represent the lowest of the minima. The second complication is that the algorithm may not converge, and the solution may oscillate wildly. Therefore, the trajectory of a point in the weight's phase space (the space of $\boldsymbol{W}$ and $d\boldsymbol{W}/dt$ or $\boldsymbol{W}$ and its increments) may be chaotic (ref. 19). Artistic adjustments are sometimes necessary to control these complications during a training procedure. The dynamics of nonlinear systems, in general, has become a topic of major research interest. The energy minimization method is the basis for the back-propagation algorithm. That algorithm was used for our alignment study and will be discussed later.

The error (eq. (7)) cannot be expected to decrease to zero for an actual training set, even when convergence is to the lowest minimum. This phenomenon can be understood in that training sets, in general, are characterized implicitly by a joint probability function or joint distribution function $P(\boldsymbol{ba})$, where the concatenation $\boldsymbol{ba}$ means that the components of $\boldsymbol{b}$ and $\boldsymbol{a}$ are the arguments of P. The use of this function accounts for the facts that (1) human generators of input/output pairs do not execute exactly the same control actions $\boldsymbol{b}$ every time they encounter essentially the same input pattern $\boldsymbol{a}$ and (2) that there are groups of scattered input patterns $\boldsymbol{a}$ which result in essentially the same control action $\boldsymbol{b}$.

Statistical analysis of training sets can be very complicated. Inputs are not necessarily independent of one another; outputs are not necessarily independent of one another. Joint probability distributions of components of $\boldsymbol{a}$ and $\boldsymbol{b}$ are not necessarily normal. Nevertheless, neural networks trained with statistically simple training sets perform in a statistically optimum manner (ref. 6). That is, the neural networks will

(1) Minimize the mean square error of equation (3)
(2) Exhibit Bayesian performance (ref. 4)
(3) Produce outputs with the maximum likelihood of being correct (ref. 4)

One way to perform a statistically simple test of a neural network is to generate training sets where the components of an input $\boldsymbol{a}$ belong to one of several multivariate normal distributions (ref. 20). The bivariate normal density for an input vector $\boldsymbol{a} = (a_1, a_2)$ is given by the equation

$$f(a_1,a_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\right.$$

$$\times\left(\frac{(a_1-u_1)^2}{\sigma_1^2} - 2\rho\,\frac{(a_1-u_1)(a_2-u_2)}{\sigma_1\sigma_2}\right.$$

$$\left.\left. + \frac{(a_2-u_2)^2}{\sigma_2^2}\right)\right] \qquad (8)$$

where σ_1 and σ_2 are the standard deviations of the two components, u_1 and u_2 are the means of the two components, and ρ is the correlation coefficient. The function f is a probability density function.

Multivariate normal distributions for any number of components are defined in reference 20. Output vectors can be defined in at least two ways. A separate output node can be associated with each distribution, or the output vector can contain a binary code of the distribution. For example, the first and fourth distributions of a four-distribution training set could be designated with output vectors (1, 0, 0, 0) and (0, 0, 0, 1) or with output vectors (0, 0) and (1, 1). Training consists of associating each input vector with its most probable distribution and executing one of the training approaches discussed earlier. Testing is based on the recognition that the normal distributions overlap. An input vector has a nonzero probability of belonging to any distribution. One test procedure would be to

(1) Use a random number generator to select the components of the input vector $\boldsymbol{a}$.
(2) Compute the probability densities that $\boldsymbol{a}$ belongs to the various distributions.
(3) Select as Bayes winner the distribution associated with the largest probability density.
(4) Use the neural network to select a distribution.
(5) Count a difference between the neural network winner and the Bayes winner as an error.
(6) Repeat this process a large number of times.
(7) Compare the measured number of errors with the number predicted from the Bayes rule.

If $P(\boldsymbol{a}/i)$ is the probability density that $\boldsymbol{a}$ is associated with the winning distribution i, then the predicted number of errors after a large number of tries is given by the sum of ratios

$$\sum_a \left[1 - \frac{P(\boldsymbol{a}|i)}{\sum_j P(\boldsymbol{a}|j)}\right] \qquad (9)$$

The first sum is over a large number of randomly selected input vectors $\boldsymbol{a}$. The Bayes rule is applied by assuming that the distributions are present with unity probability. We compute the probabilities that $\boldsymbol{a}$ is caused by each distribution, and we select as winner the distribution with the largest probability. We then use the Bayes rule to compute the probability that given $\boldsymbol{a}$, the winning distribution is i. We subtract this result from unity to calculate the probability that the losing distributions would be caused by $\boldsymbol{a}$. This process is repeated for a large number of $\boldsymbol{a}$ values. The sum rounded off is an estimate of the expected number of errors consistent with the Bayes rule.

One of the neural network development packages used (ref. 6) provides demonstration examples of this procedure for several neural network architectures. In every example, the performances of the neural networks approach that of the Bayes classifier. However, the training sets used for optical alignment will be more complex than those used for these tests. There will be correlations between different output components as well as different input components. Distributions frequently will not be normal. The networks will generate outputs not in the original training set. Nevertheless, we make the following general comments without formal mathematical proof.

Neural networks or systems of neural networks exist to perform the mappings defined by the minimization of equation (3). The most important property of these networks is their ability to generate internally, in some sense, enough degrees of freedom to meet the linear system's requirement for linearly independent exemplars. These neural networks are trained by an iterative procedure that minimizes an energy function. This energy is defined in terms of a training set of exemplars; it may be the mean square error between the exemplars and the outputs. For automated optical alignment, the training set is generated by a human operator. The training is influenced by the idiosyncratic and stochastic behavior of that operator. Hence, isolated output examples of the neural network may be erroneous. We must design a system to depend on the average behavior of a neural network and to be resistant to, or recoverable from, occasional bad directions. Because neural networks are complex nonlinear systems, hard to interpret, sometimes chaotic behavior might occur during the training process.

The use of neural networks to implement mappings appears straightforward at this point. However, once the network is trained, there is still the problem of executing the electrical and mechanical alignment. Before that problem can be tackled, however, and even before the network can be trained, the architecture of the training set itself has to be designed. What clues does a human operator use to select an alignment step? What output information is necessary to execute the alignment step? What output information is used to select a next alignment step? Clearly, knowing something about optics and optical alignment is important. Neural networks cannot replace expert knowledge entirely. The design of training sets for optical systems, and for optical alignment in particular, is discussed in the next section.

Although Maxwell's equations are linear in the electric and magnetic fields, classical optics produces many nonlinear processes. Examples are

(1) Gain saturation and feedback in optical cavities
(2) Nonlinear constitutive relations
(3) Nonlinear sensors, detectors, and recording materials
(4) Ray tracing
(5) Nonlinear practices in data handling
(6) Alignment, adaptation, and control
(7) Acousto-optic and electro-optic switching
(8) Photometry and physiological optics
(9) Nonlinear mappings from physical causes to optical effects

Nonlinear processes complicate the use of optics for studying linear phenomena. One viewpoint of neural networks is that they make nonlinear activities "transparent to the user."

The optical alignment process is a very good test case. It has the potential to involve all the nonlinearities mentioned previously. This process also tolerates the property of a trained neural network being correct on average. Because alignment is accomplished in a series of steps, a single erroneous step need not destroy the entire process.

Back-propagation-trained networks. — Back propagation refers to a variety of algorithms used to train feed-forward networks of the kind shown in figure 6. These algorithms find the minimum of an energy function expressed in weight coordinates. The method is essentially an incremental steepest-descent search. Damping in the form of so-called momentum terms, smoothing terms, or averaging is added to prevent oscillatory, or even chaotic, trajectories.

The energy function mentioned is the mean square error between the actual outputs of a feed-forward network and the corresponding training outputs. A good derivation of the unmodified algorithm is provided in reference 4. This algorithm treats weights in the output and hidden layers slightly differently. The unmodified algorithm is executed one training record at a time. The notation varies from reference to reference. The algorithm for updating the weights connecting the output layer with the adjacent hidden layer is given by the equations

$$w_{ij}^{t+1} = w_{ij}^{t} + \alpha \delta_i I_j \tag{10}$$

where

$$\delta_i = f'(s_i)\,(b_{ti} - b_i) \tag{11}$$

and where s_i is the total weighted input at output node i, I_j is the output of node j in the adjacent hidden layer, f is the nonlinear activation function, b_i is the output of output node i, b_{ti} is the training output for node i, the superscript t is an iteration number, and α is the learning rate. Note that δ_i depends on the derivative of the error. The total weighted input to output node i can include a bias term. The bias term is thought of as a weight times a unity input.

Equation (12) redefines δ for the calculation of weights for the adjacent hidden layer and subsequent hidden layers:

$$\delta_k = f'(s_k) \sum_{i=1}^{N} \delta_i W_{ik} \tag{12}$$

The δ's in the sum are from the previous layer; the weights are the old (nonincremented) weights from the layer above. Note that the learning rate α might vary from one layer to the next.

The application of the algorithm is summarized as follows:

(1) Apply an input vector.
(2) Calculate the weighted inputs s_i to the nodes of the next layer.
(3) Use the activation function to calculate the outputs I_i of the nodes.
(4) Calculate the derivatives $f'(s)$ of the activation function at each node.
(5) Continue until calculations produce an output vector $\boldsymbol{b}$.
(6) Use equations (10) and (11) and the training vector $\boldsymbol{b}_t$ to calculate δ_i and the increments in weights.
(7) Substitute the δ_i calculated in step (6) into equation (12) to calculate the δ_k in the next layer.
(8) Substitute the δ_k calculated in step (7) into equation (10) to calculate the increment in weights.
(9) Repeat steps (7) and (8) to propagate the calculation of weight increments backward through the network.
(10) Update the weights and execute the process for the next record.

This bare algorithm may work in some cases, but it is likely to lead to erratic behavior. One solution is to calculate the weight increments for all training records and to average

these increments before updating the weights. This approach constitutes a steepest-descent search for an energy involving the entire training set and is the correct approach implied by equation (5). A second solution is to add a momentum term proportional to the last increment:

$$\gamma \left(w_{ij}^{t} - w_{ij}^{t-1} \right) \tag{13}$$

The purpose of these steps is to prevent the weight space trajectory from deviating drastically from the average; the trajectory is pulled toward the trend established by the previous increments. The commercial package identified by reference 6 uses a slight modification of the momentum principle. The previous increment is multiplied by a coefficient β, and the learning coefficient α is multiplied by $1 - \beta$. This technique is called smoothing, and β is called the smoothing coefficient. The commercial package in reference 5 was used with momentum, and the commercial package in reference 6 was used with smoothing.

When it did give adequate results, the back-propagation algorithm was slow to converge. Frequently, between 10 000 and 100 000 passes through the training set were used. The parameters were changed during training. Training might start with $\alpha = 0.4$ and $\beta = 0.8$ and end with $\alpha = 0.05$ and $\beta = 0.4$. The back-propagation algorithm was able to learn individual alignments fairly well, but was unable to perform well when trained with the complete training sets.

Counter-propagation networks. – The counter-propagation network (CPN) (refs. 6 and 10) attempts to use the neural network approach to map equation (2) by table lookup. One version (ref. 6) uses a neural network to adaptively learn a table, perform lookup, and interpolate. CPN embodies its inventors' philosophy of how table lookup should be executed.

The objectives and operations of CPN can be understood by adopting a non-Euclidean geometrical viewpoint. We start with a set of records of the type $\boldsymbol{R} = (\boldsymbol{I}, \boldsymbol{T})$. The input vector $\boldsymbol{I}$ has seven elements for the spatial filter example, and the training vector $\boldsymbol{T}$ has five elements for the spatial filter example. Now, imagine that we have a seven-dimensional Cartesian coordinate system. The input vectors can be plotted in this space, and a particular grid point in this hyperspace may or may not represent a realistic input vector. This problem was discussed in the **Linear Mappings and Table Lookup** section. Figure 7 shows mechanical alignment errors for members of a training set. Now, imagine that we perform the following operations:

(1) Choose a volume in the space of $\boldsymbol{I}$ that contains the training set.

(2) Note that the volume will contain a finite number N of grid points and that these points may or may not be close to training set vectors.

(3) Now, distort the Cartesian grid in the volume.

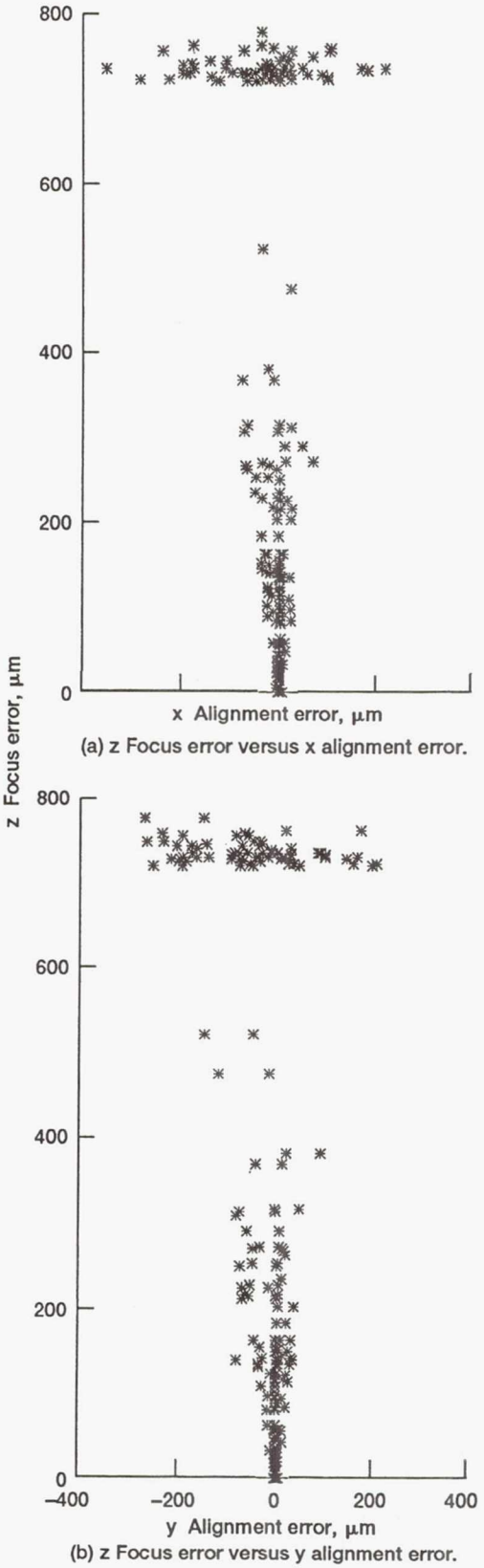


Figure 7.—Misalignment coordinates for a helium-neon training set.

(4) Pull each grid point into the midst of a group of input vector points so that

(a) An input vector belongs to a grid point if its Euclidean distance from that point is smaller than its Euclidean distance from any other point.

(b) Each of the N grid points has about the same number of input vectors.

(5) Now, average the training vectors $\boldsymbol{T}$ associated with each of the N sets of input vectors tied to a grid point and associate that average with the grid point.

Steps (1) to (6) create a table. To perform a table lookup

(1) Generate an input vector $\boldsymbol{I}$.

(2) Determine the nearest grid point by Euclidean measure.

(3) Read the training vector average entered for that grid point.

The reading in lookup step (3) is the estimate of the mapping.

This procedure allows only N levels: one for each grid point. Interpolation can be used to eliminate this quantization:

(1) Determine the n nearest grid points and their Euclidean distances S_i.

(2) Evaluate a weight factor for each grid point given by

$$e_i = \left[\frac{1}{S_i} \div \sum \frac{1}{S_j}\right]^r \qquad (14)$$

where r is usually equal to "1."

(3) Average the entries at the grid points by using the weight factors in step (2).

The interpolated value in step (3) is now the estimate of the mapping.

Essentially, CPN is a neural network that executes these operations. If we use the geometrical metaphor, each grid point is represented by a node, or neuron, in a layer called the Kohonen layer as shown in figure 8. The components of the grid point associated with a node are represented by a weight vector $\boldsymbol{W}$. Every Kohonen node is fully connected to the input layer because each node must measure its Euclidean distance $|\boldsymbol{W} - \boldsymbol{I}|$ from the input vector. The average values of the training vectors associated with each Kohonen node are stored in a second layer called a Grossberg layer. The number of nodes or neurons in a Grossberg layer equals the number of elements in the training vector $\boldsymbol{T}$ and output vector $\boldsymbol{O}$. Each Grossberg node is fully connected to the Kohonen nodes (as in fig. 8), and each connection is weighted.

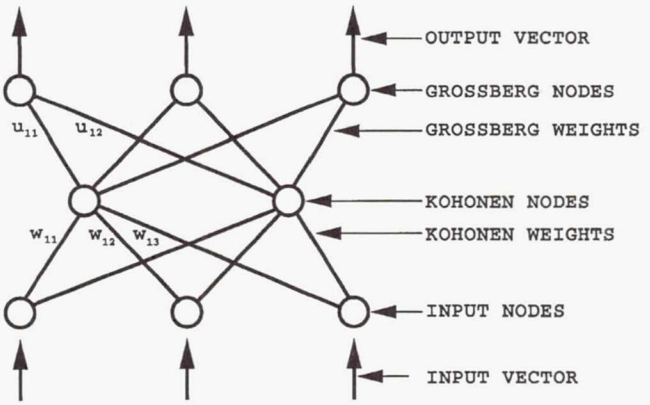


Figure 8.—Counter-propagation network (CPN).

The Grossberg layer is a linear layer. Each node has a vector $\boldsymbol{U}$ of N weight values: one weight value for each Kohonen node. The Grossberg weights corresponding to a particular Kohonen node are, of course, the averaged components of the training vectors associated with that node. In the noninterpolative mode, the outputs of the Kohonen nodes are zeros except for the winner (the least Euclidean distance node), which produces a "1." In this mode, the dot product of the vector $\boldsymbol{Z}$ of the Kohonen outputs with each weight vector $\boldsymbol{U}$ of the Grossberg nodes produces the correct output. In interpolative mode, several weighted winners are enabled; their outputs are the signals e_i defined earlier. The output of a Grossberg node is then an average of more than one of its weights.

The counter-propagation network is trained in two stages. The Kohonen layer is trained first; then the Grossberg layer is trained. Or in geometrical terms, the space of the input vectors is distorted first; then the training vectors associated with a grid point are averaged. Each training process is iterative.

The Kohonen layer is adapted to the space of input vectors. Each Kohonen node starts out with a weight vector. The grid is not necessarily Cartesian as in the earlier conceptual discussion. The Euclidean distances $|\boldsymbol{W} - \boldsymbol{I}|$ are calculated for each node, and the minimum is selected as winner. The winner is allowed to change its weights slightly to reduce the distance slightly. This fractional change is the winner's learning rate α. This process can be continued, and the space will be distorted. However, there is a problem. Some grid points might not be close enough to an input vector to ever modify the grid points' weights, and contrary to the objective, some nodes might end up with many more than their share of input vectors. This "unconscionable" result is prevented by adding a property called conscience. The winning rates of the nodes are monitored, and a node is shut down if it has too large a winning rate. Other nodes can win and adjust their weights. A node is not actually turned off. Instead, bias terms are added to the distances to increase or decrease them artificially, thereby imparting a win-rate-dependent disadvantage or advantage.

Once all the Kohonen nodes are winning at about the same rate without the need for conscience, it is time to train the Grossberg layer. In the package used for this work (ref. 6), Grossberg training occurs continuously. A fraction a of the difference between the weight and the corresponding component of the training vector is added to the weight. The coefficient a, which is called the Grossberg learning rate, is kept large while the Kohonen nodes are learning. The Grossberg nodes then effectively flip from one training value to the next. Once the Kohonen nodes have learned, the learning rate a is reduced to a very small value. The Grossberg weights for a particular Kohonen node then tend to the average of the components of the member training vectors. For CPN, the effectiveness of the learning process is determined from a mean square error.

In general, the two architectures performed equally well, or equally badly, given the same training sets. CPN is interesting because its table lookup, geometrical viewpoint is different from a viewpoint discussed previously. The previous viewpoint was that neural networks learn general mappings because the nonlinear activation function effectively transforms the training vectors into linearly independent forms. The CPN viewpoint is one of learning discriminants (ref. 4). A point that is enclosed by a volume defined by discriminant surfaces in some space is considered to be associated most likely with the properties of that volume. The properties in CPN are contained in an average training vector. Another concept is transforming a space to bring spatially separated vectors into proximity (ref. 4). In reference 4, the layers of the neural network move different input vectors with essentially the same output into the same volume of some multidimensional space.

Preclassifiers and systems of neural networks. — From a practical viewpoint, acceptable alignments of the laser-beam-smoothing spatial filter required using a preclassifier as a step in creating a system of neural networks. These systems are discussed next.

Preclassifiers use so-called unsupervised learning to group vectors into classes; that is, preclassifiers learn classes of vectors without being taught by example. This means, of course, that a classification procedure must be built into the neural network architecture from the beginning. A Euclidean preclassifier was used for this work.

The Kohonen layer in CPN is an example of a Euclidean distance preclassifier. Each class has an exemplar, and each class member has a minimum Euclidean distance from the exemplar of that class. Learning consists of setting up the exemplars for the chosen number of classes, and interrogation consists of classifying a set of vectors. The Kohonen layer adaptively moves grid points (exemplars) in the space of the input vectors until the classes (Kohonen nodes) are uniformly occupied.

For the work discussed herein, classification ended up being based mainly, but not entirely, on the states of the digital components of the input vectors. Those components are three input nodes for the control operated previously and one node for the pattern class. The control operated previously is none, Z (focus), X, or Y. The pattern class is region A or B. One system, which was trained with the helium-neon training set, contained 13 classes. Those classes are identified in table I.

The training sets constructed from the 13 classes were used to train 13 BPN's. These feed-forward networks contained a seven-node input layer for the seven-element input vector, one seven-node hidden layer, and a five-node output layer for the five-element output vector. A sigmoidal function, which was used as the nonlinear activation function, is defined by the equation

$$f(x) = \frac{1}{1 + e^{-Ax}} \tag{15}$$

where A determines the gain. As A increases, the sigmoidal function approaches the unit step function. Sigmoidal functions were used for all the BPN mappers discussed in this report.

The commercial package identified in reference 5 performed very well once it was trained. Training times for this all-software package were very long. Overnight training sessions were required for some networks. This package is useful for learning about neural networks. However, the systems discussed next were trained more rapidly with a coprocessor-based system (ref. 6).

Back-propagation-trained network and counterpropagation network using an Adaptive Resonance Theory 2 Preclassifier. — The Adaptive Resonance Theory 2 (ART2) preclassifier produced classes for the helium-neon training set that did not differ drastically from the classes produced by the Euclidean preclassifier. These classes are tabulated in

TABLE I. — CLASSES FOR EUCLIDEAN PRECLASSIFIER HELIUM–NEON TRAINING SET

Class	Control operated last	Pattern (region)	Number of training records[a]
1	NONE[b]	A	15
2	Y	A	78
3	Z	A	10
4	X	A	45
5	Z	A	28
6	Z	B	49
7	Z	A	19
8	Y	B	20
9	X	B	19
10	NONE[b]	A	16
11	Z	B	4
12	NONE[b]	A	7
13	X	A	37

[a]Each training record consists of one input vector and one output vector.

[b]Beginning of alignment.

table II. A comparison of table I with table II shows, for example, that the Euclidean preclassifier puts all the region A, previous operation = *Y* records into a single class of 78 records; whereas ART2 creates two classes where one class has 57 records and the other has 21 records. Both preclassifiers create three classes for region A, previous operation = none records. The distribution of records between the three classes is slightly different for ART2 and the Euclidean preclassifiers.

ART2 (refs. 6 and 12) monitors how well input vectors agree with those supplied in the original training set. It also embodies a significant philosophy of neural networks. This philosophy is likely to be important for optical applications; therefore, ART2 will be discussed briefly even though it was used only as a preclassifier.

TABLE II.—CLASSES FOR ART2 PRECLASSIFIER HELIUM–NEON TRAINING SET

Class	Control operated last	Pattern (region)	Number of training records[a]
1	NONE[b]	A	15
2	*Y*	A	21
3	*Z*	A	20
4	*Z*	A	31
5	*X*	A	63
6	*Y*	A	57
7	*Z*	B	53
8	*Y*	B	20
9	*X*	B	19
10	NONE[b]	A	11
11	NONE[b]	A	12
12	*X*	A	9
13	*Z*	A	6

[a]Each training record consists of one input vector and one output vector.
[b]Beginning of alignment.

The following comments might be useful in reading the literature about the Adaptive Resonance Theory (ART). The terminology used to describe ART2 in the references differs from the terminology used so far. As previously mentioned, nonstandardized terminology is a problem in this field, which only recently has been used for applications. ART is derived from ad hoc efforts to combine theories from biology and cognitive science with artificial neural networks, and its significance may be a little hard to understand by those not versed in neuronal biology. In addition, some people (such as biologists) may dislike the use of biological terms for artificial systems.

For example, in ART a layer of nodes with adjustable weights is called an adaptive filter rather than a layer of neurons, and the nodes are said to use competitive learning. Competitive learning means that only one node in a group wins the right to learn during an iteration. In that sense, a least-Euclidean-distance criterion is competitive learning. ART attempts to incorporate or emulate in a single system the formation of a properly scaled short-term memory from an input, the comparison of that short-term memory with long-term memory (bottom-up weights), the selection of a best long-term memory, the generation of an expected value for the short-term memory from long-term memory (top-down weights), a comparison between the expected and actual short-term memories (vigilance), and a decision based on that comparison. The decision based on vigilance may be (1) to accept the triggered long-term memory as correct and to modify it slightly for the new input, (2) to shut down that memory node and look for a better match, or (3) to shut down the active memory nodes and select a new node (class) in the long-term memory. This process is complicated by analog computation in several loops, by variable gain factors, and by stabilities that are established independently, with different time scales, and in different loops. In effect, recall, comparison, and learning occur continuously—but with different time constants.

ART attempts to add two features not used explicitly for this project—continuous learning or adaptation and the comparison of memory-generated expectations with inputs. The second feature appears implicitly during the alignment of a spatial filter as is explained later. In contrast, our systems of neural networks are designed to learn training sets. Once our systems are trained, they are used to respond to every vector-generated input, and they generate an output vector in response to every input vector. There is no attempt to check the validity of the output. We assume that a good training set will be inclusive; therefore, input vectors will fall in the space of the training vectors. We also assume that a slightly bad move will not be fatal, but will be corrected in the next move. These assumptions have proven, so far, to be correct for spatial filter alignments.

Suppose that there is a chance that the nature of our input vectors will change. Ideally, we would like to detect that change and to learn the appropriate input-output combination as rapidly as possible. The following approach could be considered. Train two feed-forward networks with the training set. But for the second feed-forward network, reverse the roles of the input and output vectors. That is, use the training set output vector as the input and the training set input vector as the training vector. During operations, an input vector is applied to the first network. That network generates an output vector, the output vector is applied to the second network, and a new vector appears at the system input. Then the two vectors at the system input are compared. A certain agreement is required as specified by an adjustable vigilance parameter. The system output is used for control if the agreement is good; otherwise the system is halted and learning is activated. Agreement, within the error set by the vigilance parameter, is called resonance. Learning is called adapting the resonance.

There are significant problems with this procedure. One problem is that mappings are not one-one. Consider a network that is to learn exclusive OR operation (XOR). Both (0, 0) and (1, 1) map into a zero output. Which pair should

be used in the reverse direction? The solution is to iterate the network twice. Train the reverse network to produce a (1, 1). Then, a zero output will generate that input. Now, feed the (1, 1) through again. Everything repeats on the second iteration. The processing loop must be stabilized, and there must be normalization since signal levels will vary. The nature of discrepancies must be considered in normalization. A large error in one node might be a reason to halt the network, whereas the same total error distributed over several nodes might not be a reason to halt the network.

Unsupervised learning bypasses a problem with supervised learning (which is based on input-vector, output-vector training sets). Systems training with supervised learning will fail if the training set was not designed correctly or if the training set does not contain enough examples. Unsupervised learning or pattern classification, on the other hand, is designed to accept and learn new patterns. ART is intended for pattern classification.

There are several ART architectures; ART2 consists of two versions (refs. 6 and 12). This project used ART2 to assign the training set records to classes. The vigilance parameter ρ was used to determine the number of classes. This parameter, which is in the range (0, 1), was compared with an expression containing the cosine of the angle between vectors derived from the input and the expected input vectors. Good agreement between these vectors produces an expression value close to "1." The value decreases as agreement deteriorates. A vigilance parameter that is too large produces too many classes, and a vigilance parameter that is too small produces too few classes. The helium-neon training set required $\rho = 0.98$ for 13 classes, and one argon-ion training set required $\rho = 0.99$ for 12 classes. These numbers apply to the training sets in dimensionless form.

Although ART2 was used primarily for preclassification, it did detect input vectors that deviated significantly from the training sets. Processing would halt during interrogation if the input could not be classified at the training vigilance. However, this problem did not occur often. The solution was to reduce the vigilance slightly to allow classification to proceed.

The classes established by ART2 were used to train both BPN and CPN networks. In general, a system consisting of the preclassifier and 12 or 13 BPN or CPN mappers could learn the training set adequately.

Given a trained system of artificial neural networks, the remaining task is to test that system with an actual alignment of the beam-smoothing spatial filter or with a model of that alignment. Those experimental procedures are discussed next.

The neural networks discussed here embody the ad hoc or anecdotal viewpoints of their inventors. However, all the systems of neural networks tested, regardless of the viewpoints of their creators, could learn a training set equally well, if they could learn it at all. We were unable to train any single, isolated neural network architecture adequately. Combinations of neural networks were required to learn the training sets. We suspect that the design of the training set, which combined digital and analog representations, may have been partly responsible.

Experimental Setup and Procedures

Development of Training Sets

Some previous work has been done to develop optical alignment training sets (ref. 2). The inputs of the first training sets consisted of the control last operated (*X*, *Y*, *Z*, or none), the *xy* position of the beam bright spot on a reflector, and the average brightness. The outputs consisted of the control to be operated (*X*, *Y*, *Z*, or none), the new *xy* position of the beam bright spot, and an estimate of the new brightness. The network was expected to learn the following sequence: zero *x* and *y*, reduce the focusing error *z* subject to the beam remaining on the reflector, and repeat the process until the beam brightness equals the previously known maximum. However, nets trained with this training set became locked in loops. The trained network would direct an alignment procedure that would return again and again to the same unaligned condition. The problem was that there are at least two classes of beam patterns (region A and region B in appendix A). The control to be operated and the expected behavior are different for the two classes, even though the other components of the input vector are the same.

The training set written from a human operator's exemplars must include the input clues and control actions actually employed by the human operator. There is a need to discover what items to include in the input and output vectors and when to include them.

Models of how human operators anticipate the outcome of their actions are applicable to adaptive neural networks. The difference between the anticipated outcome and the actual outcome is important in adapting the alignment procedure to different setups. The personal models of a particular human operator do not even have to be scientifically rational as long as they work. Defining personal models probably would involve an interview process similar to that used in developing the rules for an expert system.

If it is possible for the input and output vectors to include all the components in the following lists, an adequate neural-net-controlled alignment should be possible.

The input vector should contain:

(1) Previous control action and consequences
(2) Beam coordinates
(3) Beam pattern
(4) Beam brightness
(5) Expected consequences of control actions

The output vector should contain

(1) Control action to be taken
(2) Characteristics to be set (beam coordinates, beam pattern, or beam brightness)
(3) Characteristics to be estimated (again, the beam coordinates, beam pattern, or beam brightness)

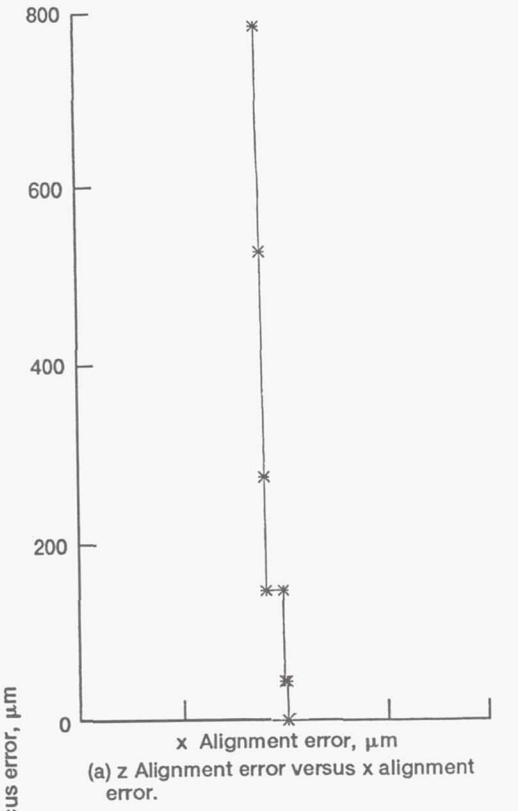
The use of neural networks that incorporate item (5) from the input vector and item (3) from the output vector is part of intelligent adaptive control and is an area of substantial ongoing research and development (ref. 21). Input item (5) requires that physical or procedural models be incorporated in the network, thereby somewhat compromising the concept of a human-trained-only system of neural networks. The work reported herein uses only a subset of the features in these lists.

Training set design for optical alignment. — A major challenge when using neural networks is the design of an appropriate training set. This requirement demands an intimate knowledge of the application. In our study, the development of an approach to designing training sets for optical alignment involved some trial and error. Only the final designs are discussed here.

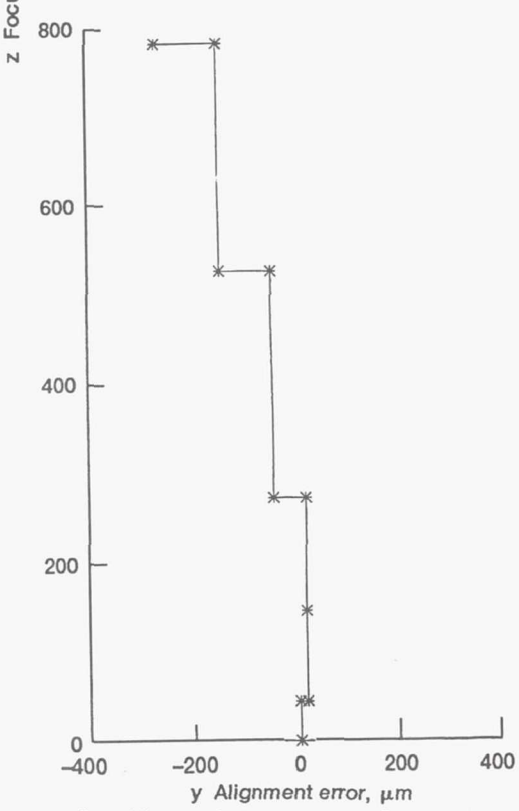
First, consider the input vector. The previous control action and consequences are represented by a three-component vector containing zeros and ones. The possible values are (0, 0, 0) for the start of an alignment (no previous control action), (1, 0, 0) for previous operation of the z-axis or focus control, (0, 1, 0) for previous operation of the x-axis control of the pinhole position, and (0, 0, 1) for previous operation of the y-axis control of the pinhole position.

The beam coordinates consisted of the x and y positions of the brightest point on the beam. These coordinates were measured relative to crosshairs drawn on the diffusely reflecting card. Initially, the pinhole was removed, and the beam was centered on these crosshairs. This step, as mentioned, is not part of the alignment procedure to be automated. Then, the pinhole was inserted, and the spatial filter was aligned carefully. The focus control was backed off about the same distance for each alignment, and the X and Y controls were set at random values. This procedure initialized an alignment ((0, 0, 0) in the previous paragraph).

Figures 7 and 9 show data for which the beam positions were not measured so carefully. Hence those data show more scatter and required more steps per alignment than subsequent training sets. It usually is very easy to measure the beam position in region A because the bright spot is typically surrounded by diffraction rings (fig. 3). The only exception is early in the alignment when the beam is faint and internal reflections can be mistaken for the main beam. Beam coordinates are sometimes difficult to define in region B. The intensity distribution can appear to be fairly uniform yet not



(a) z Alignment error versus x alignment error.



(b) z Alignment error versus y alignment error.

Figure 9.—Misalignment coordinates for one alignment.

symmetrical. It can have a wispy texture. Sometimes, the measurement of a beam coordinate is little more than a guess. Most of the alignment steps occur in region A, but the complexity of region B makes the spatial filter an excellent benchmark component for testing neural networks.

We characterized the beam pattern by only 1 bit: a zero for region A and a "1" for region B. There was little point in using a more complex characterization in the absence of a machine vision system. Future work will probably require more complex characterizations, at least in region B, and perhaps to handle internal reflections at the beginning of region A.

The beam brightness was measured as follows. The 1-cm^2 sensor of a power meter was centered on the bright spot, and the average power in microwatts per centimeter squared was measured after each control action. The base-10 logarithm of this power was used as the beam brightness in a somewhat rudimentary attempt to emulate an operator's visual response. The brightness was then renormalized easily for different laser powers and for different distances of the reflecting card.

There was no attempt to anticipate the consequences of the control actions. It is important to ask to what extent human operators use a personal mental model of the alignment procedure to anticipate the consequences of their actions. As a rule, the outputs of that model should be inputs to the neural network. Expected consequences of control actions should be investigated as an input for future research.

Now, consider the output vector. The control action to be taken is again represented by a three-component vector. The choices are (0, 0, 0) for no action (the appropriate choice when the alignment is complete), (1, 0, 0) for operation of the z-axis or focus control, (0, 1, 0) for operation of the pinhole's x-axis position control, and (0, 0, 1) for operation of the pinhole's y-axis position control. The only characteristics to be set are the x and y coordinates of the beam's bright spot. This choice is certainly adequate for region A; region B alignments eventually might require setting some beam pattern parameters. Typically, the human operator zeros x or y when the x or y control action is selected. Both x and y are allowed to increase when the focus or z-axis control action is selected. They are allowed to increase until the distance of the beam bright spot from the center reaches a maximum permissible value; that maximum is the output value when selecting the focus control.

Beam brightness is the only characteristic to be estimated. That characteristic was fed back as an input during simulated alignments of the spatial filter, which required a model-generated choice for the beam pattern.

The design of a training set record is then summarized as follows. There is a seven-component input vector consisting of three previous-control-action nodes, two beam-coordinate nodes, one beam-pattern node, and one beam-brightness node. There is a five-component output vector consisting of three control-action nodes, one beam-position node, and one beam-brightness node. A training record then contains 12 elements, and training records are sequenced to form an alignment. Alignments are executed from random starting positions to form a training set.

Two slight modifications of this design were investigated: (1) a training set where the change in brightness was used in place of the brightness and (2) a training set where the beam brightness and beam coordinates were made dimensionless.

Dimensionless training sets.—Human operators can align spatial filters for different gaussian beam parameters, laser powers, distances of the viewing card, microscope objectives, pinhole diameters, and mechanical designs of the spatial filter assembly. The dimensionless training set is a very primitive attempt to emulate this generality. Learning the general procedure by example is the ultimate goal.

The beam incident on the viewing card is approximately a single spatial and single temporal eigenmode of the electromagnetic field. Generic eigenmodes are represented in terms of a limited number of variables, and these variables can be made dimensionless. Unfortunately, the spatial mode changes during the alignment process. Dividing the alignment process into two regions called A and B in appendix A greatly simplifies describing alignments, but changes still are particularly noticeable in region B.

There is not a unique dimensionless training record. The region A and region B models were used to guide the creation of the following particular example. The dimensionless variables differ between regions A and B. Control identifications, such as (0,1,0) are unaffected, of course. The 1-bit identification of the beam pattern also is unaffected. The position coordinates X, Y are replaced by

$$\frac{Xf}{(Zw_s)} \text{ and } \frac{Yf}{(Zw_s)} \text{ in region A}$$

and by

$$\frac{\lambda f^2 X}{\left(w_s^2 DZ\right)} \text{ and } \frac{\lambda f^2 Y}{\left(w_s^2 DZ\right)} \text{ in region B}$$

The dimensionless coordinates for region A follow from the geometrical optics interpretation discussed in appendix A. The coordinate is essentially the angular location (X/Z, Y/Z) of the bright spot normalized with respect to the polar angle (w_s/f) of the $1/e^2$ edge of the beam. The square of this coordinate also is the argument of the exponential in equation (A13) in appendix A after applying the region A assumptions to that equation. The geometrical optics interpretation does not depend on the wavelength; therefore, the wavelength does not appear in the definition of the dimensionless coordinates for region A. The coordinates refer to a single point (the

position of the beam bright spot); therefore, D also does not appear in the definition. The pinhole diameter D does affect the spread of the diffraction ring pattern about the bright spot.

The dimensionless coordinates for region B are based on equation (B6) in appendix B, but are not as easy to define because the shape of this mode varies significantly. The dimensionless groupings in the exponential and in the terms following the exponential are different. The position of a bright spot, if one occurs, will be determined by the bracketed terms. This suggests using the definition

$$\frac{X_z}{w_f^2 d_\lambda} = \left(\frac{\lambda f^2 X}{w_s^2 DZ}, \frac{\lambda f^2 Y}{w_s^2 DZ} \right) \tag{16}$$

Brightness is a sum of the logarithms of factors such as those in equation (A13) or (A15) in appendix A or in equation (B6) in appendix B. Hence, normalization (or renormalization) is accomplished by subtracting these logarithms expressed in dimensionless variables. The first step is to express the beam power P in terms of the maximum axial intensity at the reflecting card. The intensity averaged over the $1/e^2$ diameter also could be used, but the effect is constant and not important in dimensional analysis. Equation (A2) is used for this purpose by substituting

$$Z \text{ in place of } \delta_z$$

$$R \Rightarrow \infty$$

$$w' \Rightarrow Zw_s / f$$

The result is given by

$$\sqrt{\frac{2P}{\pi}} = \frac{Zw_s}{f} \sqrt{I_{\max}} \tag{17}$$

This result is substituted in equation (A13) or (A15). These equations are affected by the mechanical misalignment variables that are unknown during the formation of a training set. In particular, w' depends on δ_z. If we choose $\delta_z = 0$ to form a dimensionless parameter, the quantity to be subtracted from the brightness elements in the experimental training sets is then given by

$$log \left[W_f^4 \left(\frac{\pi^2 d_\lambda^2}{4} \right)^2 I_{\max} \right]$$

where

$$d_\lambda = \frac{D}{\lambda} \text{ and } W_f = \frac{w_s}{f} \tag{18}$$

These transformations are done on the experimental training set; then the transformed training set is used to train the neural network. The neural network is interrogated by transforming an input vector, presenting it to the network, and performing the inverse transformation on the output vector. In addition, scaling transformations can be performed to use the full dynamic range available from the networks. These activities were performed by C-language functions, and the results were stored internally.

Experimental Procedure for Neural-Net-Directed Alignments

Apparatus and software. — The experimental setup consisted of the beam-smoothing spatial filter (figs. 1 and 2) with two combinations of the microscope objective and pinhole; a helium-neon and an argon ion laser together with mirrors to direct the laser beams to the spatial filter; a diffusely reflecting card with crosshairs for observing and centering the patterns of the laser beams; a power meter for measuring the average beam power; several experimentally generated training sets, which were produced by Kenneth E. Weiland (one of the authors); commercially supplied neural network hardware and software (refs. 5 to 6), and an AT microcomputer. Figure 10 is a photograph of the experimental setup. Most of the experimental work was done with the second package (ref. 6) of hardware and software. That package included a coprocessor that was installed in a PC/AT microcomputer.

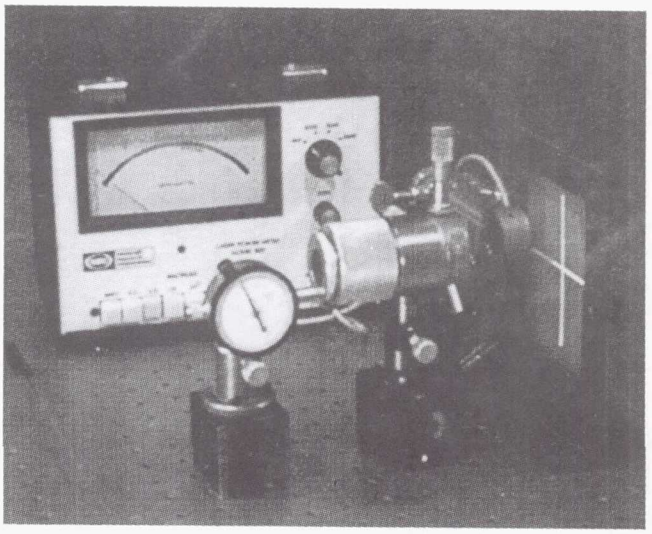


Figure 10.—Typical experimental setup for acquiring training sets for the spatial filter alignment.

The package could be accessed at three levels: a menu-driven level, a C-language function level, and a compiler level. The menu-driven level supported two algorithms for learning mappings: BPN and CPN. The C-language function level supported 17 algorithms. It also could operate systems of neural networks by passing the outputs of one network to the inputs of the next. The compiler could create new neural-network architectures. All three levels were used, but most of the data reported herein were collected using the C-language function level.

The package identified in reference 5 also was used to create systems of neural networks. This package relied entirely on software, and it required long training times. Nevertheless, its trained systems of neural networks performed as well as those of the other package. The reference 5 package also offered a different viewpoint. Recall that the nonlinear activation functions of networks internally generate the degrees of freedom needed to handle the mappings. In contrast, the reference 5 package can perform nonlinear operations on the input vector and supply the results of these nonlinear operations as additional inputs. In some cases, a mapping can be learned by a single-layer network when this procedure is used (ref. 4).

The experimental procedure is straightforward: select a format for the training set members or records, execute a large number of alignments recording the training set record at each step, select a neural network or combination of neural networks to learn the training set, execute the training algorithm for the neural networks while monitoring some measure of learning success, note the difficulty and time for training, and test the effectiveness of training by using the neural networks to direct the alignment of a spatial filter. The experimenter also must learn the architectures and programming languages for the neural computers.

This report discusses neural-network-*directed,* alignments. In other words, the trained neural network was used to pass alignment instructions to a human operator. The human operator executed the instructions and then passed a vector of inputs back to the neural network. The neural network then generated another instruction. This experiment is not a neural-network-*controlled* alignment of the spatial filter. Neural-network-controlled alignments of the spatial filter, with actuator-driven degrees of freedom, are, of course, the real goal of the overall research and development effort.

Neural-network-directed alignments of a spatial filter.— Two kinds of experiments were conducted: neural-network-directed alignments of actual spatial filters and neural-network-controlled alignments of a model spatial filter. The configuration of the neural network and of its parameters were determined by the designer who monitored its learning progress.

Neural-network-directed alignments were performed on the same setups used to create the alignment records for training. The same person who created the training sets also executed these alignments as directed by the system of neural networks.

In preparation for the experiment, the spatial filter was aligned carefully. Then, the focus was backed off typically about 700 to 900 μm, and X and Y were randomly turned. The misadjustments of X and Y were not so large as to make the beam invisible.

The beam characteristics and previous history were specified, measured, and used to create input vectors as described in the *Training set design for optical alignment* section. The input vector was then relayed to the computer operator who then entered the input vector. The AT computer contained the neural network coprocessor (ref. 6) and its software (refs. 5 and 6). Generally, the system of neural networks responded immediately with an output vector. Occasionally, an input vector could not be classified at the same level of vigilance as was used to prepare the training sets. Then it was necessary to reduce the vigilance slightly to force classification and routing of the input vector to a mapper. The output instructions were then relayed to the person performing the alignment as follows:

(1) The control to be operated (Z (focus), X, Y, or none)
(2) If control = X, the new x position
(3) If control = Y, the new y position
(4) If control = Z (focus), the new distance from the center

The control to be operated was selected by rounding off the first three components of the output vector. The largest value ≥ 0.5 was rounded up to 1.0. A tie was settled by rounding Z first and X second. All other values were equated to 0.0. The predicted brightness was recorded for comparison with the actual brightness. The predicted brightness was not used by the person performing the alignment.

The focus adjustment sometimes required a modification of these output instructions. This modification decreased the value of the experiment because it required interpretation after the training was completed. When Z (focus) was adjusted, the beam could not always be forced to the distance indicated by the output vector. Hence, action on step (4) of the procedure was modified as follows:

(4a) If possible, the predicted distance was set.
(4b) If the brightness decreased too much, the distance at which the beam was barely visible was set.
(4c) If the brightness increased without the predicted increase in distance, the distance at which the beam stopped brightening was set.

The person performing the alignment executed the instructions and then measured a new input vector. Sometimes, photographs were recorded of the card reflected beam. This

procedure was continued until the system of neural networks issued a "halt" or "all-done" output vector defined by (0, 0, 0) in the first three components.

The person executing the instructions designated the beam as region A or B and measured the position of the beam bright spot relative to crosshairs.

The systems of neural networks also were used to align a model spatial filter as discussed next.

Neural-network-directed alignments of a model spatial filter. – Some versions of the software contained an option to complete an alignment from an initial input vector. This option was used to test the self-consistency of the system of networks; that is, it determined whether a system of networks would proceed to an aligned state on the basis of only a string of its own input and output vectors. In contrast, the alignment of an actual spatial filter, as discussed in the previous section, provided continuous corrections through remeasurements of the input vector. A second application of alignment completion was used to provide rapid visualization of an alignment with the procedures discussed in appendix B.

For the self-consistency test, the output vector, the input vector, and a very coarse model were used to create a new input vector. The following procedure was used:

(1) The first three components of the new input vector were created from the first three components of the output vector by rounding off as defined in the previous section.

(2) If X was selected, the new X value and the old Y value were selected for the next two components.

(3) If Y was selected, the old X value and the new Y value were selected for the next two components.

(4) If Z (focus) was selected, then the old X and Y values were multiplied by the new distance from the center divided by the old distance from the center, and the results were used as the next two components.

(5) The beam pattern, given by the next component, was determined from the brightness. Region B was defined to occur with an output brightness greater than or equal to 3.39.

(6) The brightness in the new input vector was taken, of course, from the last component of the output vector.

(7) The new input vector was used to plot the new beam position and profile.

(8) The new input vector was sent to the system of networks for another iteration.

(9) The process was halted when the first three components of the output rounded off to (0, 0, 0).

In step (4) there is a danger of division by zero. However, the original training set and the model training set involved sufficient errors in zeroing that division by zero never occurred.

Results and Discussion

Comparison of Training the Back-Propagation-Trained Network and the Counter-Propagation Network

ART2 was used to preclassify the argon ion training set into 12 classes. These classes were then taught to both the back-propagation-trained network (BPN) and the counter-propagation network (CPN). The entire training set was rendered dimensionless, and the individual classes were rescaled prior to training the mappers. Rescaling places elements of the vectors in the range (-0.9, 0.9). The two architectures were equally effective at learning these training sets as can be seen by examining table III. Table III tabulates mean square errors for each class for BPN and CPN. The mean square error is measured after training by performing one more pass through the training set.

Table IV lists the types of input vectors characterizing the different classes of table III. The largest mean square errors appear for the larger classes where X or Y was the control last operated. Although the input vectors in a class are characterized by the same values of the first three components and of the sixth component, brightness and positions can vary substantially. The output vectors can vary in the control to be operated, the brightness, and the position. Mappers are limited by a statistically best performance. The fact that two completely different forms of mappers achieve the same performance indicates that this limit has been reached.

BPN and CPN learned the class training sets equally well; they can be compared according to other criteria. Both networks were trained with 12 000 iterations (passes through the

TABLE III. – COMPARISON OF BPN WITH CPN: DIMENSIONLESS AND RESCALED ARGON–ION TRAINING SETS

Class	Number of training records[a]	Mean square error	
		BPN	BPN
1	50	0.006006	0.004297
2	50	.079269	.072757
3	51	.120310	.123502
4	49	.008547	.002877
5	40	.062840	.084559
6	39	.009437	.000000
7	41	.098409	.090904
8	3	.000000	.000000
9	5	.000001	.000000
10	5	.082420	.081007
11	6	.000002	.000000
12	2	.000000	.000000

[a]Each training record consists of one input and one output vector.

TABLE IV.—CHARACTERIZATION OF TRAINING CLASSES FROM TABLE III: DIMENSIONLESS AND RESCALED ARGON-ION TRAINING SETS

Class	Number of training records[a]	Mean square error	
		BPN	BPN
1	50	NONE	A
2	50	*Y*	A
3	51	*X*	A
4	49	*Z*	A
5	40	*X*	A
6	39	*Z*	B
7	41	*Y*	A
8	3	*Z*	B
9	5	*X*	B
10	5	*Z*	B
11	6	*Y*	B
12	2	*Z*	B

[a]Each training record consists of one input and one output vector.

training sets). The training of CPN was faster, but it required more frequent adjustments of parameters. The CPN parameters (Kohonen learning rate, a parameter called the bias multiplier, and the Grossberg learning rate) (ref. 6) were adjusted six times during training. The BPN parameters were adjusted three times. CPN must be large for large training sets, whereas the size of BPN does not change. All the BPN nets had 7 input nodes, 14 nodes in one hidden layer, and 5 output nodes. The CPN network used to learn the third class contained 2548 bytes. The total sizes of the network files for the third class contained 1744 bytes for BPN and 4260 bytes for CPN. One advantage of CPN is its large number of Kohonen nodes: accuracy increases as the number of active Kohonen nodes increases. CPN requires somewhat arbitrary choices of the number of Kohonen nodes to use for interpolation and of the value of the interpolation exponent *r*. The relationship between accuracy and the number and sizes of hidden layers in BPN is difficult to discern.

Alignment Tests With Neural Network Systems

The test of a trained system of neural networks was whether that system could direct the alignment of a laser-beam-smoothing spatial filter. The results of such tests are discussed in the following section.

The objective for this work is to automate the alignment and operation of optical measurement systems in inaccessible aerospace environments. The only acceptable test, therefore, is to demonstrate alignments of optical components. The following alignment test is representative; it shows some problems and indicates clearly the required direction for future work.

The system tested was used to generate tables III and IV. Figure 11 is a photographic record of the alignment test (ref. 2) that was recorded with the version of the system that used CPN as a mapper. A neural network system consists of a preclassifier and about 10 mappers. CPN was used with the interpolation technique discussed in the description of the CPN earlier. Up to six Kohonen nodes were allowed to participate in determining the output where the interpolation exponent was $r = 2$. These values were chosen by trial and error tests on the original training set. BPN mappers, which also successfully directed the alignment of the spatial filter, do not require these adjustments.

Table V lists the neural net alignment in training set form. There are three differences between the form of this table and the form of the typical training set. First, the beam pattern is listed as "A" or "B" for region A or B rather than as "0" or "1." Second, the brightness in the output vector generally differs from the brightness in the subsequent input vector. The output brightness is predicted, and the input brightness is measured. Third, the distance predicted for an adjustment of Z(focus) may not equal the distance achieved because of limitations in the apparatus as discussed in an earlier section.

Table V can be compared with figure 11, but the following comments must be kept in mind. Figure 11 contains an imaging reversal: right is interchanged with left and top is interchanged with bottom. In addition, saturation makes it difficult to show the bright spot; therefore, the beam may appear to be off center when the bright spot is actually centered. The center of an image must be overexposed to bring out the ring pattern. The region A versus region B judgment was made by the human operator actually executing the alignment instructions. The same human operator recorded the training sets as stated previously. The person operating the computer and relaying the instructions was in a different room.

Figure 11 is to be read in television raster fashion from left to right and top to bottom. The first three frames of figure 11, representing the first three lines in table V, clearly are region A patterns. Multiple diffraction rings are visible in all three frames. The broken appearances of the diffraction rings may be caused by spatial variations in the sensitivity of the film used to record the photographs. They also could be anomalous diffraction effects from dirt or pinhole imperfections, because the simple theory applies to a perfectly circular, undamaged, clean pinhole.

The operator classified the next two frames as region A frames also. However, the photograph shows them to be region B frames. Nevertheless, the system of neural networks could order essentially correct moves in spite of the

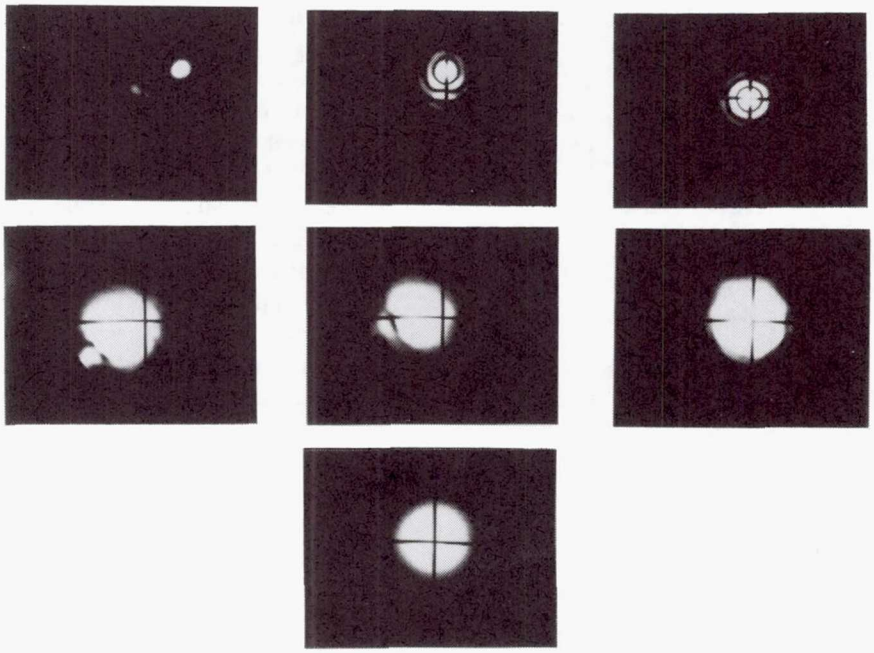


Figure 11.—Photographic record of neural-net-directed alignment of spatial filter.

TABLE V.—NEURAL–NETWORK–DIRECTED ALIGNMENT IN TRAINING RECORD FORM FOR ARGON–ION TRAINING SET AND CPN MAPPERS

Input					Output		
Control (Z, X, Y)	Position		Pattern (region)	Bright-ness	Control (Z, X, Y)	Position (x, y, or distance)	Bright-ness
	x	y					
0 0 0	-20.0	-14.0	A	-0.744	0 1 0	0.0	0.560
0 1 0	.0	-14.0	A	.602	0 0 1	.0	1.338
0 0 1	.0	.0	A	1.415	1 0 0	16.0	1.810
1 0 0	6.0	.0	A	3.415	0 1 0	.0	3.504
0 1 0	.0	.0	A	3.613	1 0 0	.0	3.585
1 0 0	.5	.0	B	3.602	0 1 0	.0	3.556
0 1 0	.0	.0	B	3.633	0 0 0	.0	3.574

pattern error. A system of neural networks trained with an earlier training set did not incorporate pattern information, yet it could direct complete alignments in many cases. That system, however, would occasionally get stuck between two states and oscillate back and forth between them, with some states in between. It turned out that the second state was one of a pair of states that had identical input vectors. One state of this pair produced a region A pattern and the other a region B pattern. Treating these patterns as identical caused the system of networks to direct the alignment back to an earlier state rather than to an aligned state. Hence, we decided to incorporate 1 bit of pattern information, leading eventually to the region A, region B theory.

There are two points to be made from the preceding discussion. The first is that a serial alignment process appears to be robust in the sense that occasional errors and bad decisions do not destroy the whole process. The second is really the main point of this whole report: neural nets are intended to learn by example. They are adapted to learn craftsmanship rather than academic knowledge. The need to make judgments on the basis of physical theories must be regarded as a defect, and the only way to avoid this defect is to acquire a training set for a complete system. A complete system for the spatial filter would consist of a machine vision system based, for example, on a charge-coupled device (CCD) camera and frame grabber together with electromechanical actuators. A complete system would record a training set of input-output vectors without substantial intervention by the operator. The operator's only contribution would be skilled example.

Incidently, a model trained neural network also was fairly successful at directing the alignment of the laser-beam-smoothing spatial filter.

Table VI contains an alignment sequence in which BPN was the mapper. The alignment with BPN started from the same point as with CPN and followed nearly the same path.

TABLE VI. – NEURAL–NETWORK–DIRECTED ALIGNMENT IN TRAINING RECORD FORM FOR ARGON–ION TRAINING SET AND BPN MAPPERS

Input					Output		
Control (Z, X, Y)	Position		Pattern (region)	Bright-ness	Control (Z, X, Y)	Position (x, y, or distance)	Bright-ness
	x	y					
0 0 0	-20.0	-14.0	A	-0.744	0 1 0	0.0	0.442
0 1 0	.0	-14.0	A	.602	0 0 1	.0	1.354
0 0 1	.0	.0	A	1.415	1 0 0	16.0	1.892
0 0 1	6.0	.0	A	3.415	0 1 0	.0	3.584
0 1 0	.0	.0	A	3.613	1 0 0	.0	3.594
1 0 0	.5	.0	B	3.602	0 1 0	.0	3.570
0 1 0	.0	.0	B	3.633	0 0 0	.0	3.570

The result in the previous section was that BPN and CPN learned the 12 sets of training vectors equally well. The most noteworthy observation is that line 4 of table IV contains an experimental error. In this line, Y was entered as the last control operated rather than Z, as indicated by the previous output vector. The BPN network, which was erroneously consulted, still directed the correct move. The argon-ion trained network was trained to precisely zero x or y whenever one or the other occurred with a nonzero value. The helium-neon training set is different. That training set was originally constructed by measuring errors in the mechanical drives for each alignment step. Crosshairs were not used to zero the beam coordinates.

Figures 7 and 9 were constructed from this original training set. Later, the beam coordinates were recovered by making a second pass through the training set. The precision of that procedure was limited, because of the nonlinear effects in the mechanical drives. The helium-neon training set gives results closer to human alignment, yet it is less efficient. Another point is that CPN learned the helium-neon training set better than BPN. With CPN, we can choose any number of Kohonen nodes, up to the maximum. In effect, we can have one node for each training entry. Generally, we need Kohonen nodes equal to one-third or more of the training entries.

Concluding Remarks

An important conclusion drawn from this work is that a neural-network-controlled alignment process should be trained and tested in its entirety. Ideally, the environment for training and testing will be the environment for the final application. Training and testing should be nonverbal. The alignment expert should view the light pattern on a monitor attached to the machine vision system used by the network. The alignment expert should perform alignment actions via the actuators used by the network, and the training set should be recorded automatically. Then the trained system should be tested by how well it completes alignments without human intervention. Any procedure that programs a network with weights learned in a laboratory is a weak procedure.

The neural-network-directed alignments of the spatial filter described herein do not meet this standard. The training sets were designed to be recorded by the alignment expert, and they incorporated the human expert's interpretation of the beam pattern. The alignment tests also required human interpretation and human translation of the output vectors into mechanical actions. The discussion of tables V and VI shows how imperfect interpretations can be.

Nevertheless, the neural-network-directed alignments produced some important conclusions and some motivations for additional work. The approach mandated in the first paragraph of this section is one conclusion. Another conclusion is that neural-network-controlled sequencers are very robust in the sense that they tolerate mistakes. The alignment path has an excellent chance of recovering from an erroneous move and proceeding to an aligned state. This single property is a good reason for continuing the development of neural networks for alignment.

This work also points out the importance of adaptive systems. The long-range goal for such systems is the control of adaptive optics. A more realistic near-term project is to develop and test a system to correct for misalignment induced by the environment. Vibrationally induced misalignment is an example. This project is consistent with the program objective of automating alignments in remote, harsh, and dangerous environments and can probably be demonstrated with commercially available equipment. The objective is quite different from the spatial filter alignment discussed in this report. The alignment of the spatial filter requires proceeding in steps from a misaligned state to an aligned state. The adaptive system would need to learn, detect, and correct for transitions from aligned to misaligned states.

The adaptive resonance theory (ART) discussed in this report is one system for using unsupervised learning to detect and classify new states of misalignment. Learning to respond correctly to these new states of misalignment is the essence of adaptation. This type of learning is one step removed

from the training algorithms already discussed. Essentially, the system of neural networks must learn how to learn the correct response to the misalignment. The textbook approach is to have a separate network that contains and implements a model of learning. A primitive approach might consist of interrupting the experiment when a new state is detected and classified. (As discussed in this report, ART2 does this automatically via the vigilance parameter.) A human operator would then teach the networks a correct response, and the experiment (actually the network development) would continue. As mentioned, some people consider the creation of full adaptation to be the most important area for research. The use of neural networks to compensate for environmentally caused misalignment in a component used in the field would be an ideal demonstration project.

This is a very appropriate point to recite some conclusions about the current resources for implementing neural networks. The current software and hardware are slow, and they have comparatively small memories (in relation to their human operator counterparts). The project suggested in the previous paragraph might require or benefit from some customized, front-end hardware for rapid acquisition and classification of misalignment states. It is a mistake, however, to claim that present neural network demonstrations are not real because they use digital computers. The neural network architectures are quite real, but the digital computer cannot take advantage of the ability of groups of identical neurons to be updated at the same time. The digital computer essentially updates these neurons one at a time. The conceptual difficulty is easily eliminated by recognizing that true neural networks also update neurons one at a time, if a short enough time interval is selected. Certainly, neural network applications will benefit from the development of parallel hardware.

The comparison of BPN and CPN suggests some interesting conclusions about resources and technologies. CPN is fully equivalent to table lookup when the number of Kohonen nodes or neurons equals the number of entries. The Grossberg layer performs weighted interpolation. CPN performed as well as BPN for the relatively small training sets that were used for the experiments.

The advantage of CPN is that the number of weighted connections increases in proportion to the size of the training set, whereas the number of weighted connections in BPN increases in proportion to a power of the size of the input vector. This effect did not create a problem for the seven-element input vectors used for the spatial filter alignments. It certainly will create a problem for input vectors containing tens of thousands of elements, as discussed in connection with the use of neural networks for processing optical data. One conclusion is that CPN is the network of choice for small data sets. A more important conclusion is that table lookup may be a significant, superior alternative to neural networks for a long time. Large memories are becoming inexpensive and algorithms such as CPN are available for learning and organizing tables. The role of table lookup in mapping and directing the alignment of spatial filters was discussed with figures 7 and 9. Effective competition from existing technologies is always a factor in developing new technologies.

Neural networks should have a significant role in processing optical data. A nonlinear network has the ability to reconstitute a compressed data set. In effect, feed-forward networks have unlimited internal degrees of freedom that can store and resupply missing data, provided that the data can be reconstituted by some definite rule. The neural network, of course, learns the rule implicitly, transparently, and by example. It will simply classify the inputs in the Bayesian sense, at best, if it cannot discover a rule. Neural networks offer an efficient way to study data compression. The payoff can be enormous in the aerospace field. A current system uses holograms that must be recorded through windows. The hologram must be processed; then the information is measured comparatively slowly from 29 different views. A data compression to three views, followed by reconstitution, would allow holography to be replaced by high-speed electronic interferometry. The retention of all the views, but with a few measurements per view, would permit the use of fiber-optic interferometry, thereby eliminating the need for windows and solving the optical access problem in the aerospace field.

Finally, a fully automated spatial filter alignment must be demonstrated. Neural networks are indeed the expert systems of craftsmanship. They learn by example. Only the complete system can avoid the need for the verbal intercourse that spoils this example learning. A person with an optical systems background will feel very comfortable with neural networks. It works very well for pattern-based processes, is well constructed for research in adaptive optics, and has potential for processing optical data. The development of supporting technologies and competition from existing technologies will set the timetable for applications of neural networks.

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135, May 1993

Appendix A — Theory and Models for Beam-Smoothing Spatial Filter

The derivation of the Fresnel diffraction theory (ref. 22) of the beam-smoothing spatial filter is straightforward, but the theory is hard to use for calculations. This difficulty should not be surprising because the patterns observed in the step or two before alignment is achieved are complex. In this appendix, the diffraction theory is used to generate easy-to-handle approximate models of the alignment of a spatial filter. This appendix has three sections: the general diffraction theory of the spatial filter, a model of the alignment process when the filter is out of focus, and a model of the alignment process when the filter is focused or nearly focused.

Fresnel Diffraction Theory

This discussion (fig. 12) assumes the use of a gaussian laser beam (ref. 23) that is coaxial with the optical axis of the spatial filter assembly. A thin lens of focal length f replaces the microscope objective commonly used to focus the laser beam, and aberrations are ignored. The pinhole, or spatial filter, is assumed to be in the xy plane at $z = 0$. The center of the circular pinhole is assumed to be misaligned with (x,y) coordinates δ_x and δ_y, and the center of the beam waist is assumed to be misfocused with the z-coordinate δ_z. The result of a successful alignment is to zero or nearly zero these coordinates. The remaining parameters are the pinhole diameter D, the distance (z-coordinate) Z at which the beam pattern is observed or measured, and the laser-beam characteristics.

The laser-beam characteristics are defined as follows. The laser beam has a wavelength λ and a $1/e^2$ radius w_s at the lens aperture, the beam waist appears at the focus at distance f from the lens, and the vignetting effect of the lens is ignored. The beam waist then has a $1/e^2$ radius given by

$$w = \frac{\lambda f}{\pi w_s} \tag{A1}$$

The beam power is called P.

We obtain a formal expression for the beam pattern by mathematically propagating the beam to the pinhole, multiplying the pattern by the pinhole aperture function, and then mathematically propagating the apertured beam to the viewing plane at distance Z. Constant phase factors are not retained because they cancel in evaluating the intensity.

The field at the pinhole plane $z = 0$, minus constant phase factors, is given by

$$u(\rho) = \sqrt{\frac{2P}{\pi}}\ \frac{1}{w'}\ e^{-A\rho^2} \tag{A2}$$

where

$$A = \left(\frac{1}{w'}\right)^2 - \frac{j\pi}{\lambda R} \tag{A3}$$

$$w' = w\sqrt{1 + \left(\frac{\lambda\delta_z}{\pi w^2}\right)^2} \tag{A4}$$

$$R = \delta_z\left[1 + \left(\frac{\pi w^2}{\lambda\delta_z}\right)^2\right] \tag{A5}$$

$$\rho^2 = x^2 + y^2 \tag{A6}$$

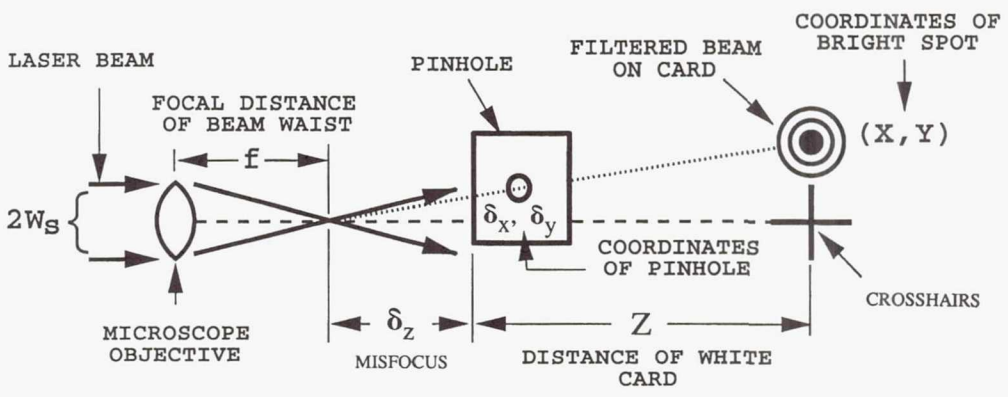


Figure 12.—Simplified diagram of spatial filter setup.

This field (eq. (A2)) is multiplied by a circular aperture function $C(x - \delta_x, y - \delta_y)$ that is centered on the pinhole and is unity inside the pinhole and zero outside the pinhole. The Fraunhofer diffraction integral is then used to represent the field at the observation plane. Capital letters X, Y, Z represent the coordinates of the observation plane. The scalar field in the observation plane is then given by

$$U(X) = \sqrt{\frac{2P}{\pi}}\ \frac{1}{w'}\ \frac{1}{\lambda Z}\int e^{-A\rho^2} \times C(\rho - \delta)e^{-j(2\pi/\lambda z)(x\cdot\rho)}d\rho$$

where

$$\rho = (x,y) \tag{A7}$$

$$\delta = (\delta_x,\delta_y)$$

$$X = (X,Y)$$

Equation (A7) is evaluated in part by applying the shift and convolution theorems of Fourier transformations (ref. 22). The result is given by

$$U(X) = \sqrt{\frac{2P}{\pi}}\ \frac{1}{w'}\ \frac{1}{\lambda Z}\ \frac{\pi D}{2A}\int e^{-j2\pi F'\cdot\delta} \times \frac{J_1(\pi F'D)}{F'}\ e^{-(\pi^2/A)(F - F')^2}\,dF' \tag{A8}$$

where

$$F = \frac{X}{\lambda Z}$$

$$F = |F|$$

The symbol J_n represents a Bessel function of the first kind of order n. Integrals are evaluated over the entire domain of the variable of integration.

Equation (A8) can be evaluated formally with the assistance of a table of integrals (ref. 24). The first step is to integrate with respect to the polar angle. The result is given by

$$U(X) = \sqrt{\frac{2P}{\pi}}\ \frac{1}{w'}\ \frac{1}{\lambda Z}\ \frac{\pi^2 D}{A}\ e^{-(\pi^2/A)F^2} \times \int_0^\infty e^{-(\pi^2/A)F'^2} I_0(BF')J_1(\pi F'D)\ dF' \tag{A9}$$

where

$$B = \sqrt{\frac{4\pi^4}{A^2}\ F^2 - 4\pi^2\delta^2 - j\ \frac{8\pi^3}{A}\ d\cdot F}$$

The symbol I_n represents a modified Bessel function where $I_0(x) = J_0(jx)$. The exponential preceding the integral sign represents the beam profile in the absence of the filter. The integral itself is interpreted as a beam-profile, beam-position modification factor; it represents the main effect of filter misalignment.

The integral in equation (A9) can be evaluated formally, and the result is given by

$$U(X) = \sqrt{\frac{2P}{\pi}}\ \frac{1}{w'}\ \frac{1}{\lambda Z}\ \frac{\pi D^2}{4}\ e^{-(\pi^2/A)F^2} \sum_{m=0}^{\infty}\frac{1}{m!}\left(\frac{B^2A}{4\pi^2}\right)^m \times F\left(-m,\ -m;\ 2;\ -\left(\frac{\pi D}{B}\right)^2\right) \tag{A10}$$

where

$$F\left(-m,\ -m;\ 2;\ -\left(\frac{\pi D}{B}\right)^2\right) = 1 + \sum_{i=1}^{m}\frac{m!m!}{i!(i+1)!(m-1)!}-\left(\frac{\pi^2D^2}{B^2}\right)^i \tag{A11}$$

Equations (A1) to (A11) represent the diffraction theory of the process of aligning a spatial filter. It is more convenient in this report to use approximate models based on the theory. The model discussed next represents the alignment process when the filter is well out of focus.

Region A Model—Filter Out of Focus

Most alignment steps occur when the spatial filter is substantially out of focus. For example, for typical values,

$$f = 8{,}300\ \mu\text{m}$$

$$\lambda = 0.5145\ \mu\text{m}$$

$$w_s = 1{,}750\ \mu\text{m}$$

the beam $1/e^2$ diameter will exceed a typical pinhole diameter of 10 μm when $\delta_z = 20$ μm. Because the misfocus can be as large as 1000 μm at the start of an alignment process, the beam diameter can exceed the pinhole diameter over 98 percent of the focus adjustment range.

A model of this region can be generated with equation (A7). First the equation is transformed to the center of the pinhole, and then the remaining quadratic exponential under the integral is assumed to deviate negligibly from 1.0. The resulting integral is easily evaluated to yield the following result:

$$U(X) = \sqrt{\frac{2P}{\pi}}\ \frac{1}{w'}\ \frac{1}{\lambda Z}\ \frac{D}{2}\ e^{-A\delta^2}\ \frac{J_1\left(\pi D\left|\frac{jA\delta}{\pi} - \frac{X}{\lambda Z}\right|\right)}{\left|\frac{jA\delta}{\pi}\ \frac{X}{\lambda Z}\right|} \tag{A12}$$

The model is especially simple if the imaginary terms in the argument of the Bessel function and in the denominator of equation (A12) are ignored. The result is given by

$$U(X) = \sqrt{\frac{2P}{\pi}}\ \frac{1}{w'}\ \frac{1}{\lambda Z}\ \frac{\pi D^2}{4}\ e^{-A\delta^2}\ \frac{2J_1\left(\frac{\pi D}{\lambda Z}\left|\frac{\delta Z}{R} - X\right|\right)}{\frac{\pi D}{\lambda Z}\left|\frac{\delta D}{R} - X\right|} \tag{A13}$$

Choosing $R = \delta_z$ produces an especially simple result. The center of the diffraction ring pattern is then located at

$$X = \frac{\delta_x}{\delta_z} Z$$

$$Y = \frac{\delta_y}{\delta_z} Z \tag{A14}$$

This center is exactly that determined from geometrical optics by drawing a line from the center of the waist, through the center of the pinhole, to the observation plane.

Equations (A13) and (A14) define the region A model for a beam that is out of focus. It is equally easy to produce a model when the beam is nearly focused. That model is discussed next.

Region B Model—Filter Focused or Nearly Focused

The model is developed directly from equation (A9). As stated, the exponential in front of the integral is the beam profile with the pinhole removed, and the integral can be thought of as a shape modification factor. In region B, I_0 is assumed to vary slowly in comparison with J_1. It is removed from the integral with an argument evaluated at the first maximum of J_1, which occurs at

$$\pi DF' = 1.8$$

The integral remaining is approximately $DA/4\pi$, and the overall result is given by the equation

$$U(X) = \sqrt{\frac{2P}{\pi}}\ \frac{1}{w'}\ \frac{1}{\lambda Z}\ \frac{\pi D^2}{4}\ e^{-(\pi^2/A)F^2}\ I_0\left(\frac{1.8B}{\pi D}\right) \tag{A15}$$

The I_0 in equation (A15) constitutes a complex shape function and also permits the intensity maximum to occur off axis as is sometimes observed in region B. Equation (A15) is the region B model.

The models, like the diffraction theory, are approximate. Criteria for switching from region A to region B are somewhat arbitrary. Nevertheless, neural networks trained with these models perform well.

Appendix B – Visualization of Spatial Filter Alignment

A person who aligns a spatial filter looks at and is guided by the diffuse reflection of the filtered and diverging laser beam. The subjective appearance of that beam depends on the beam shape and beam parameters discussed in appendix A. The appearance also depends on the beam power, the properties of the diffuser, room lighting, and nonideal effects such as escaping reflections that originate inside the filter assembly.

Similar comments apply to any other visualization method such as a camera and a monitor. Definite, although occasionally erroneous, decisions must be made on the basis of such subjective visualizations. This alignment example requires that the observer (human or mechanical) decide on a value of beam position, on whether the beam pattern resembles the region A model or region B model, and on some estimate of brightness. Even the choice of the position of the diffuser is a somewhat arbitrary decision.

The interpretation of visual information is a good subject for neural networks. The work reported in this report used visualization for demonstrations only; standardization was used to reduce arbitrariness.

Standardization

The standardization is defined as follows:

(1) The beam pattern shall be presented in a 32- by 32-pixel format.

(2) The unfiltered beam shall be centered in the 32- by 32-pixel window when the spatial filter (pinhole) is removed.

(3) The overall magnification of the beam recording system shall be set such that the 32- by 32-pixel window is $4w'$ on a side, where w' is the $1/e^2$ radius of the unfiltered laser beam.

(4) The value of a pixel shall be based on the logarithm of irradiance (intensity) or a corresponding photometric quantity. The radius w' in item (3) can be estimated by replacing δ_z in equation (A4) with Z, which is the distance from the aligned pinhole to the diffuser.

Example of Visualization Process

There are many ways to display the beam pattern in a manner consistent with the standard described in the previous section. The following method was used to display a schematic representation of the filtered laser beam during neural-network-directed alignments of the spatial filter.

A C-language function was created to be used with EGA graphics software. The function was designed either to plot an externally supplied 32- by 32-pixel array or to compute and plot such an array from a seven-element input vector of the kind discussed in the Development of Training Sets section.

Inputs to the function were (1) a pointer to the seven-element input array, (2) a pointer to the array of pixels, (3) the $1/e^2$ radius w_s of the laser beam at the lens, (4) the wavelength λ of the laser beam, (5) an estimate of the maximum logarithm of intensity encountered during the alignment, (6) an estimate of the minimum logarithm of intensity encountered during the alignment, (7) the distance Z to the diffuser, (8) the diameter D of the pinhole, (9) the focal length f of the lens, and (10) a flag indicating whether the array of pixels was to be supplied externally or calculated internally.

Element 5 of the input vector (in C-language, the first element is indexed as zero) decided the form of the internal calculation if one was called for. Equation (A13) in appendix A was adopted if element 5 = 0, indicating a region A pattern of rings. Positions on the diffuser were expressed in units of Z, and the $1/e^2$ radius w_s was made dimensionless in units of f, as in

$$w_f = \frac{w_s}{f} \tag{B1}$$

The pinhole diameter D was made dimensionless in units of wavelength λ, as in

$$d_\lambda = \frac{D}{\lambda} \tag{B2}$$

The logarithm of intensity at the center of the pattern was estimated as element 6. In the experiments, element 6 was computed from an average intensity measured over a fairly large detector size. The main effect of ignoring that fact is equivalent to adding the same offset to all region A values. The Bessel function was computed with a commercially available software package. The 32-by-32 pixels were stepped out horizontally and vertically from $-2w_f$ to $2w_f$ in the normalized x and y position variables.

A much more complex region B pattern must be calculated when element 5 = 1. An already simplified expression for this pattern is given by equation (A15) in appendix A. This expression was simplified further to provide a convenient expression for visualization. However, the dimensionless groupings defined by equations (B1) to (B4) are applicable to the full region A and region B models. These groupings also suggest the training of neural networks using dimensionless inputs. A dimensionless position and a dimensionless xy misalignment are defined by the equations

$$X_z = \frac{X}{Z} \tag{B3}$$

and

$$\delta_D = \frac{\delta}{D} \tag{B4}$$

Equation (A15) is replaced by a proportionality with the first two terms of a small-argument approximation of the Bessel function substituted. That proportionality is given by

$$U(X) \propto \exp\left(-\frac{\pi^2}{A}F^2\right)\left[1 + \frac{1}{4}\left(\frac{1.8B}{\pi D}\right)^2\right] \tag{B5}$$

where the assumptions near focus are

$$R \rightarrow \infty$$

$$A \rightarrow \frac{1}{w^2}$$

$$w' \rightarrow w$$

The proportionality expressed in the dimensionless variables of equations (B1) to (B4) is given by

$$U(X_z) \propto \exp\left(-\frac{X_z^2}{w_f^2}\right) \times \left[1 + 0.328\left(\frac{1}{w_f^4 d_\lambda^2}X_z^2 - \pi^2\delta_D^2 - \frac{j2\pi}{w_f^2 d_\lambda}\,\delta_D \cdot X_z\right)\right] \tag{B6}$$

The intensity for both regions A and B is proportional to the field times its complex conjugate:

$$I(X_Z) \propto U^*(X_Z)\, U(X_Z)$$

The position of the beam bright spot (elements 3 and 4) and the logarithm of intensity (element 6) are combined with equations (B6) and (B7) to derive the components of δ. These components of δ are substituted in equation (B6), and the array of pixels is calculated as for region A.

The array of pixels was represented on an EGA monitor by associating a 4-by-5 array of screen pixels with each of the calculated pixels. The four vertical pixels and the five horizontal pixels correspond to the aspect ratio of the screen. The actual emulation of the beam on the monitor is somewhat arbitrary—the color (red, green, blue, or yellow) is chosen from the wavelength. The logarithm of intensity at the beam bright spot ranges between the minimum and maximum values supplied to the *C*-function; however, an actual camera would have a variable iris whose setting might range from fully opened at the minimum brightness to nearly closed at the maximum brightness. The iris effect was inserted as a brightness-dependent offset to the logarithm of the intensity at the bright spot. The brightness of a calculated pixel was represented by the number of screen pixels illuminated in the 4-by-5 representation of the calculated pixel. The iris correction and the order of illumination of the screen pixels were adjusted by trial and error in an effort to create the visual effect of an actual alignment of the spatial filter.

Final Comments on Visualization

The *C*-function just described takes many seconds to compute and display a beam profile, in spite of the simplifications, standardization, and artistic license mentioned. The fact that a beam profile can be represented with dimensionless variables is the most important finding of appendix B. A neural network trained with dimensionless inputs is intended to work for any spatial filter and laser beam combination.

Appendix C – Symbols

Lengths, including wavelength, are normally expressed in millimeters and occasionally in micrometers. Vectors and matrices are denoted by bold faced type.

A	area of iris aperture, or coefficient in sigmoid function (eq. (15)), or complex coefficient in gaussian exponent (eq. (A3)), or in reference to region A model
$\boldsymbol{a}$	input vector (eq. (2))
B	coefficient (eq. (A9)), or in reference to region B model
$\boldsymbol{b}$	output vector (eq. (2))
$\boldsymbol{b}_t$	training vector (eq. (7))
$C(\,)$	circular aperture function
D	diameter of pinhole (spatial filter)
d_λ	D/λ
E	mean square error (eq. (3))
e_i	weight for Kohonen node i when CPN is used with interpolation
$\boldsymbol{F}$	$(X/\lambda Z, Y/\lambda Z)$
$F(a, b; c; x)$	hypergeometric function
f	effective focal length of microscope objective
$f(\,)$	mapping function (eq. (2))
$\boldsymbol{I}$	another expression for input vector
$I(\,)$	irradiance or intensity function
I_j	output of node j
$I_{\max}$	maximum irradiance in unfiltered beam as measured at white card
$I_0(\,)$	modified Bessel function
$J_n(\,)$	Bessel function of the first kind and order n
N	number of training records
$\boldsymbol{O}$	another symbol for output vector
P	probability density, or beam power
$P(a/i)$	conditional probability or probability density (eq. (9))
R	radius of curvature of gaussian beam wave front (eq. (A5))
$\boldsymbol{R}$	training record $(\boldsymbol{I}, \boldsymbol{T})$ where $\boldsymbol{I}$ is an input vector and $\boldsymbol{T}$ is a training vector
r	exponent used in formula for calculating weight factors when CPN is used in interpolation mode
S_i	Euclidean distance between vector and grid point i
s	weighted input at node
s_i	total weighted input at node i
$\boldsymbol{T}$	training vector
t	time, or iteration index
$\boldsymbol{U}$	vector of Grossberg weights in CPN
$U(\,)$	scalar field
u	mean value (eq. (8))
$u(\,)$	scalar field at pinhole (eq. (A2))
$\boldsymbol{W}$	linear mapping or transformation matrix (eq. (1)) or matrix of neural network weights (eq. (4))
w	gaussian beam waist (eq. (A1))
w'	gaussian beam $1/e^2$ radius (eq. (A4))
w_f	w_s/f
w_{ij}	weight at node i for signal from node j
w_s	$1/e^2$ radius of laser beam at microscope objective
X	Cartesian x-coordinate of beam bright spot, or x-position control on spatial filter assembly

$\boldsymbol{X}$	(X, Y)
$\boldsymbol{X_Z}$	$(X/Z, Y/Z)$
x	x-position in pinhole plane
Y	Cartesian y-coordinate of beam bright spot, or y-position control on spatial filter assembly
y	y-position in pinhole plane
Z	Cartesian z-coordinate, or distance from pinhole (spatial filter) to white card, or focus control
$\boldsymbol{Z}$	Kohonen outputs in CPN
α	learning rate (eq. (10))
β	smoothing coefficient; replaces momentum coefficient in one version of back-propagation algorithm
γ	momentum coefficient (back-propagation algorithm)
δ	(δ_x, δ_y)
δ_D	δ/D
δ_i	error used in back-propagation algorithm (eq. (10))
δ_x	x displacement of pinhole from optical axis
δ_y	y displacement of pinhole from optical axis
δ_z	focus error
λ	wavelength of laser beam
ρ	correlation coefficient (eq. (8)), or vigilance parameter used by ART2, or radial distance in plane of pinhole (eqs. (A2) and (A6))
ρ	(x, y)
σ	standard deviation (eq. (8))
$\|\ \|$	absolute value or magnitude of enclosed terms
$< >$	statistical expectation value of enclosed terms

References

1. Maren, A.J.: A Logical Topology of Neural Networks. Proceedings of the Second Workshop on Neural Networks: Academic/Industrial/NASA/Defense–WNN–AIND91, M.L. Padgett, ed., SPIE Vol. 1515, Society for Computer Simulation International, San Diego, CA, 1991, pp. 29–36.
2. Decker, A.J.: Self Aligning Optical Measurement Systems. Appl. Opt., vol. 31, no. 22, Aug. 1, 1992, pp. 4339–4340.
3. Rumelhart, D.E.; and McClelland, J.L.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations. MIT Press, Cambridge, MA, 1986.
4. Pao, Y.H.: Adaptive Pattern Recognition and Neural Networks. Addison-Wesley, Reading, MA, 1989.
5. N_NET 210 vended by AI WARE, Inc. 11000 Cedar Avenue, Cleveland, OH 44106.
6. ANZA PLUS vended by Hecht-Nilsen, Inc. 5501 Oberlin Drive, San Diego, CA 92121–1718.
7. Fuzzy Logic Comparator vended by Micro Devices. 5695 B Beggs Road, Orlando, FL 32810–2603.
8. Seasholtz, R.G.; Oberle, L.G.; and Weikle, D.H.: Optimization of Fringe-Type Laser Anemometers for Turbine Engine Component Testing. AIAA Paper 84–1459, 1984 (Also NASA TM–83658, 1984).
9. Rumelhart, D.E.; Hinton, G.E.; and Williams, R.J.: Learning Internal Representations by Error Propagation. Report ICS-8506, Institute for Cognitive Science, California University, San Diego, CA, 1985.
10. Hecht-Nielson, R.: Counterpropagation Networks. Appl. Opt., vol. 26, no. 23, Dec. 1, 1987, pp. 4979–4984.
11. Pao, Y.H.: Adaptive Pattern Recognition and Neural Networks. Addison-Wesley, Reading, MA, 1989, pp. 214–217.
12. Carpenter, G.A.; and Grossberg, S.: ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns. Appl. Opt., vol. 26, no. 23, Dec. 1, 1987, pp. 4919–4930.
13. Beaumont, R.A.; and Ball, R.W.: Introduction to Modern Algebra and Matrix Theory. Holt, Rinehart, Winston, New York, 1954.
14. Padgett, M.L.; and Roppel, T.A.: WNN91 Perspectives on Neural Networks. Proceedings of the Second Workshop on Neural Networks: Academic/Industrial/NASA/Defense–WNN–AIND91, M.L. Padgett, ed., SPIE Vol. 1515, Society for Computer Simulation International, San Diego, CA, 1991, pp. 3–6.
15. McClelland, J.L.; and Rumelhart, D.E.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 2: Psychological and Biological Models. MIT Press, Cambridge, MA, 1986.
16. Koch, C.; and Segev, I.: Methods in Neuronal Modeling: From Synapses to Networks. MIT Press, Cambridge, MA, 1989.
17. Moon, F.C.: Chaotic Vibrations: An Introduction for Applied Scientists and Engineers. Wiley, New York, 1987.
18. Natterer, F.: The Mathematics of Computerized Tomography. Wiley, New York, 1986.
19. Manausa, M.E.; and Lacher, R.C.: Chaos and the Step-Size Dilemma in the Back-Prop Learning Algorithm. Proceedings of the Second Workshop on Neural Networks: Academic/Industrial/NASA/Defense–WNN–AIND91, M.L. Padgett, ed., SPIE Vol. 1515, Society for Computer Simulation International, San Diego, CA, 1991, pp. 153–160.
20. Thomas, J.B.: An Introduction to Statistical Communication Theory. Wiley, New York, 1969, pp. 52–64.
21. White, D.; and Sofge, D.: NSF Workshop on Aerospace Applications of Neurocontrol. Proceedings of the Second Workshop on Neural Networks: Academic/Industrial/NASA/Defense–WNN–AIND91, M.L. Padgett, ed., SPIE Vol. 1515, Society for Computer Simulation International, San Diego, CA, 1991, pp. 523–527.
22. Goodman, J.W.: Introduction to Fourier Optics. McGraw-Hill, New York, 1968.
23. Koechner, W.: Solid State Laser Engineering. Springer-Verlag, New York, 1976, pp. 176–178.
24. Gradshteyn, I.S.; and Ryzhik, I.M.: Table of Integrals, Series, and Products; Corrected and Enlarged Edition Prepared by A. Jeffrey. Academic Press, New York, 1980.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1993	Technical Paper

4. TITLE AND SUBTITLE

Neural–Network–Directed Alignment of Optical Systems Using the Laser–Beam Spatial Filter as an Example

5. FUNDING NUMBERS

WU–590–21–11

6. AUTHOR(S)

Arthur J. Decker, Michael J. Krasowski, Kenneth E. Weiland

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–7524

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TP–3372

11. SUPPLEMENTARY NOTES

Responsible person, Arthur J. Decker, (216) 433–3639.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 35

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This report describes an effort at NASA Lewis Research Center to use artificial neural networks to automate the alignment and control of optical measurement systems. Specifically, it addresses the use of commercially available neural network software and hardware to direct alignments of the common laser-beam-smoothing spatial filter. The report presents a general approach for designing alignment records and combining these into training sets to teach optical alignment functions to neural networks and discusses the use of these training sets to train several types of neural networks. Neural network configurations used include the adaptive resonance network, the back-propagation-trained network, and the counter-propagation network. This work shows that neural networks can be used to produce robust sequencers. These sequencers can learn by example to execute the step-by-step procedures of optical alignment and also can learn adaptively to correct for environmentally induced misalignment. The long-range objective is to use neural networks to automate the alignment and operation of optical measurement systems in remote, harsh, or dangerous aerospace environments. This work also shows that when neural networks are trained by a human operator, training sets should be recorded, training should be executed, and testing should be done in a manner that does not depend on intellectual judgments of the human operator.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Neural networks; Computer software; Optics	36
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

National Aeronautics and
Space Administration
Code JTT
Washington, D.C.
20546-0001

Official Business

Penalty for Private Use, $300

BULK RATE
POSTAGE & FEES PAID
NASA
Permit No. G-27

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return